Aligning Transformers with Weisfeiler–Leman

Luis Müller [1] **Christopher Morris** [1]

Abstract

Graph neural network architectures aligned with the k-dimensional Weisfeiler–Leman (k-WL) hierarchy offer theoretically well-understood expressive power. However, these architectures often fail to deliver state-of-the-art predictive performance on real-world graphs, limiting their practical utility. While recent works aligning graph transformer architectures with the k-WL hierarchy have shown promising empirical results, employing transformers for higher orders of k remains challenging due to a prohibitive runtime and memory complexity of self-attention as well as impractical architectural assumptions, such as an infeasible number of attention heads. Here, we advance the alignment of transformers with the k-WL hierarchy, showing stronger expressivity results for each k, making them more feasible in practice. In addition, we develop a theoretical framework that allows the study of established positional encodings such as Laplacian PEs and SPE. We evaluate our transformers on the large-scale PCQM4Mv2 dataset, showing competitive predictive performance with the state-of-the-art and demonstrating strong downstream performance when fine-tuning them on small-scale molecular datasets.

1. Introduction

Message-passage graph neural networks (GNNs) are the de-facto standard in graph learning (Gilmer et al., 2017; Kipf & Welling, 2017; Scarselli et al., 2009; Xu et al., 2019a). However, due to their purely local mode of aggregating information, they suffer from limited expressivity in distinguishing non-isomorphic graphs in terms of the 1-dimensional Weisfeiler–Leman algorithm (1-WL) (Morris et al., 2019; Xu et al., 2019a). Hence, recent works (Azizian & Lelarge, 2021; Maron et al., 2019a; Morris et al., 2020; 2023; 2022) introduced *higher-order* GNNs, aligned with the k-dimensional Weisfeiler–Leman (k-WL) hierarchy for graph isomorphism testing (Cai et al., 1992), resulting in more expressivity with an increase in the order $k > 1$. The k-WL hierarchy draws from a rich history in graph theory (Babai, 1979; Cai et al., 1992; Grohe, 2017; Weisfeiler & Leman, 1968), offering a deep theoretical understanding of k-WL-aligned GNNs. In contrast, graph transformers (GTs) (Glickman & Yahav, 2023; He et al., 2023; Ma et al., 2023; Müller et al., 2024; Rampášek et al., 2022; Ying et al., 2021) recently demonstrated state-of-the-art empirical performance. However, they draw their expressive power mostly from positional or structural encodings (PEs in the following), making it challenging to understand these models in terms of an expressivity hierarchy such as the k-WL. In addition, their empirical success relies on modifying the attention mechanism (Glickman & Yahav, 2023; Ma et al., 2023; Ying et al., 2021) or using additional message-passing GNNs (He et al., 2023; Rampášek et al., 2022).

To still benefit from the theoretical power offered by the k-WL, previous works (Kim et al., 2021; 2022) aligned transformers with k-IGNs, showing that transformer layers can approximate invariant linear layers (Maron et al., 2019b). Crucially, Kim et al. (2022) designed *pure transformers*, requiring no architectural modifications of the standard transformer and instead draw their expressive power from an appropriate *tokenization*, i.e., the encoding of a graph as a set of input tokens. Pure transformers provide several benefits over graph transformers that use message-passing or modified attention, such as being directly applicable to innovations for transformer architectures most notably Performers (Choromanski et al., 2021) and Flash Attention (Dao et al., 2022) to reduce the runtime or memory demands of transformers, as well as the Perceiver (Jaegle et al., 2021) enabling multi-modal learning. Unfortunately, the framework of Kim et al. (2022) does not allow for a feasible transformer with provable expressivity strictly greater than 1-WL due to an $O(n^6)$ runtime complexity and the requirement of 203 attention heads resulting from their alignment with IGNs. This poses the question of whether a more feasible hierarchy of pure transformers exists.

[1]Department of Computer Science, RWTH Aachen University, Germany. Correspondence to: Luis Müller <luis.mueller@cs.rwth-aachen.de>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

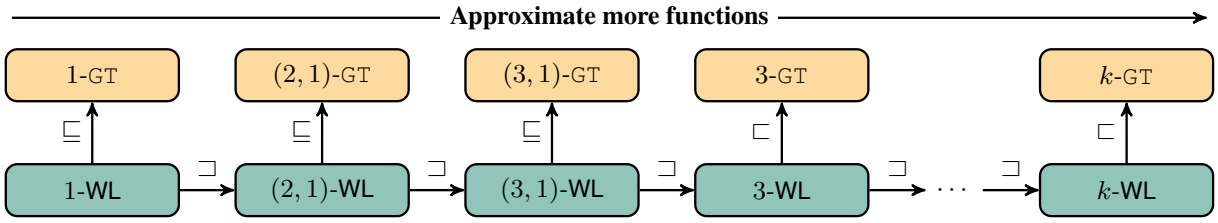

Figure 1: Overview of our theoretical results, aligning transformers with the established k-WL hierarchy. Forward arrows point to more powerful algorithms or neural architectures. $A \sqsubset B$ ($A \sqsubseteq B$, $A \equiv B$)—algorithm A is strictly more powerful than (as least as powerful as, equally powerful as) B. The relations between the boxes in the lower row stem from Cai et al. (1992) and Morris et al. (2022).

Present work Here, we offer theoretical improvements for pure transformers. Our guiding question is

> *Can we design a hierarchy of increasingly expressive pure transformers that are also feasible in practice?*

In this work, we will make significant progress to answering this question in the affirmative. First, in Section 3, we improve the expressivity of pure transformers for every order $k > 0$, while keeping the same runtime complexity as Kim et al. (2022) by aligning transformers closely with the Weisfeiler–Leman hierarchy. Secondly, we demonstrate the benefits of this close alignment by showing that our results directly yield transformers aligned with the (k, s)-WL hierarchy, recently proposed to increase the scalability of higher-order variants of the Weisfeiler–Leman algorithm (Morris et al., 2022). These transformers then form a hierarchy of increasingly expressive pure transformers. We show in Section 4, that our transformers can be naturally implemented with established node-level PEs such as the Laplacian PEs in Kreuzer et al. (2021). Lastly, we show in Section 5 that our transformers are feasible in practice and have more expressivity than the 1-WL. In particular, we obtain very close to state-of-the-art results on the large-scale PCQM4Mv2 dataset (Hu et al., 2021) and further show that our transformers are highly competitive with GNNs when fine-tuned on small-scale molecular benchmarks (Hu et al., 2020a). See Figure 1 for an overview of our theoretical results and Table 1 for a comparison of our pure transformers with the transformers in Kim et al. (2021) and Kim et al. (2022).

Related work Many graph learning architectures with higher-order Weisfeiler–Leman expressive power exist, most notably δ-k-GNNs (Morris et al., 2020), SpeqNets (Morris et al., 2022), k-IGNs (Maron et al., 2019b;c), PPGN (Azizian & Lelarge, 2021; Maron et al., 2019a), and the more recent PPGN++ (Puny et al., 2023). Moreover, Lipman et al. (2020) devise a low-rank attention module possessing the same power as the folklore 2-WL. Bodnar et al. (2021) propose CIN with an expressive power of at least 3-WL. However, while theoretically intriguing, higher-order GNNs often fail to deliver state-of-the-art performance on real-world problems (Azizian & Lelarge, 2021; Morris et al., 2020; 2022), making theoretically grounded architectures less relevant in practice.

Graph transformers with higher-order expressive power are Graphormer-GD (Zhang et al., 2023) as well as the higher-order graph transformers in Kim et al. (2021) and Kim et al. (2022). However, Graphormer-GD is less expressive than the 3-WL (Zhang et al., 2023). Further, Kim et al. (2021) and Kim et al. (2022) align transformers with k-IGNs, and, thus, obtain the theoretical expressive power of the corresponding k-WL. However, they do not empirically evaluate their transformers for $k > 2$. For $k = 2$, Kim et al. (2022) propose *TokenGT*, a pure transformer with a $n + m$ tokens per graph, where n is the number of nodes and m is the number of edges. This transformer has 2-WL expressivity, which, however, is the same as 1-WL expressivity (Morris et al., 2023). From a theoretical perspective, the transformers in Kim et al. (2022) still require impractical assumptions such as $bell(2k)$ attention heads, where $bell(2k)$ is the $2k$-th bell number (Maron et al., 2019a),[1] resulting in 203 attention heads for a transformer with a provable expressive power strictly stronger than the 1-WL. In addition, Kim et al. (2022) introduces special encodings called node- and type identifiers that are theoretically necessary but, as argued in Appendix A.5 in (Kim et al., 2022), not ideal in practice. For an overview of the Weisfeiler–Leman hierarchy in graph learning; see Morris et al. (2023).

2. Background

We consider *node-labeled graphs* $G := (V(G), E(G), \ell)$ with n nodes and without self-loops and without isolated nodes, where $V(G)$ is the set of *nodes*, $E(G)$ is the set of *edges*, and $\ell \colon V(G) \to \mathbb{N}$ assigns an initial *color* or *label*

[1] For example, $bell(2 \cdot 3) = 203$, $bell(2 \cdot 4) = 4\,140$, $bell(2 \cdot 5) = 115\,975$. In comparison, GPT-3, a large transformer with 175B parameters, has only 96 attention heads (Brown et al., 2020).

to each node. For convenience of notation, we always assume an arbitrary but fixed ordering over the nodes such that each node corresponds to a number in $[n]$. Further $\mathbf{A}(G) \in \{0,1\}^{n\times n}$ denotes the *adjacency matrix* where $\mathbf{A}(G)_{ij} = 1$ if, and only, if nodes i and j share an edge. We also construct a *node feature matrix* $\mathbf{F} \in \mathbb{R}^{n\times d}$ that is *consistent* with ℓ, i.e., for nodes i and j in $V(G)$, $\mathbf{F}_i = \mathbf{F}_j$ if, and only, if $\ell(i) = \ell(j)$. Note that, for a finite subset of $\mathbb{N}$, we can always construct $\mathbf{F}$, e.g., with a one-hot encoding of the initial colors. We call pairs of nodes $(i,j) \in V(G)^2$ *node pairs* or 2-tuples. For a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, whose i-th row represents the embedding of node $v \in V(G)$ in a graph G, we also write $\mathbf{X}_v \in \mathbb{R}^d$ or $\mathbf{X}(v) \in \mathbb{R}^d$ to denote the row corresponding to node v. Further, we define a *learnable embedding* as a function, mapping a subset $S \subset \mathbb{N}$ to a d-dimensional Euclidean space, parameterized by a neural network, e.g., a neural network applied to a one-hot encoding of the elements in S. Moreover, $||\cdot||_{\mathrm{F}}$ denotes the Frobenius norm. Finally, we refer to some transformer architecture's concrete set of parameters, including its tokenization, as a *parameterization*. See Appendix C for a complete description of our notation. Further, see Appendix E.1 for a formal description of the k-WL and its variants.

3. Expressive power of transformers on graphs

Here, we consider the (standard) *transformer* (Vaswani et al., 2017), a stack of alternating blocks of *multi-head attention* and fully-connected *feed-forward networks*; see Appendix D for a definition.

To align the above transformer with the k-WL, we will provide appropriate input tokens $\mathbf{X}^{(0,k)}$ to the standard transformer for each $k \geq 1$ and subsequently show that then the t-th layer of the transformer can simulate the t-th iteration of some k-order Weisfeiler–Leman algorithm. To gradually introduce our theoretical framework, we begin with tokenization for $k = 1$, obtaining a transformer with the expressive power of at least the 1-WL and afterward generalize the tokenization to higher orders k.

3.1. Transformers with 1-WL expressive power

A commonly used baseline in prior work (Müller et al., 2024; Rampášek et al., 2022) is to employ a standard transformer with Laplacian PEs (Kreuzer et al., 2021). Here, we start by characterizing such a transformer in terms of 1-WL expressivity and introduce our theoretical framework. Concretely, we propose a tokenization for a standard transformer to be at least as expressive as the 1-WL. We first formalize our tokenization and then derive how 1-WL expressivity follows. Let $G = (V(G), E(G), \ell)$ be a graph with n nodes and feature matrix $\mathbf{F} \in \mathbb{R}^{n\times d}$, consistent with ℓ. Then, we initialize n token embeddings $\mathbf{X}^{(0,1)} \in \mathbb{R}^{n\times d}$ as

$$\mathbf{X}^{(0,1)} \coloneqq \mathbf{F} + \mathbf{P}, \tag{1}$$

where we call $\mathbf{P} \in \mathbb{R}^{n\times d}$ *structural embeddings*, encoding structural information for each token. For node v, we define

$$\mathbf{P}(v) \coloneqq \mathsf{FFN}(\deg(v) + \mathrm{PE}(v)), \tag{2}$$

where $\deg\colon V(G) \to \mathbb{R}^d$ is a learnable embedding of the node degree, $\mathrm{PE}\colon V(G) \to \mathbb{R}^d$ is a node-level PE such as the Laplacian PE (Kreuzer et al., 2021) or SPE (Huang et al., 2024), and $\mathsf{FFN}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a multi-layer perceptron. For the PE, we require that it enables us to distinguish whether two nodes share an edge in G, which we formally define as follows.

Definition 1 (Adjacency-identifying)**.** Let $G = (V(G), E(G), \ell)$ be a graph with n nodes. Let $\mathbf{X}^{(0,1)} \in \mathbb{R}^{n\times d}$ denote the initial token embeddings according to Equation (1) with structural embeddings $\mathbf{P} \in \mathbb{R}^{n\times d}$. Let further

$$\tilde{\mathbf{P}} \coloneqq \frac{1}{\sqrt{d_k}} \mathbf{P}\mathbf{W}^Q \big(\mathbf{P}\mathbf{W}^K\big)^T,$$

where $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d\times d}$. Then $\mathbf{P}$ is *adjacency-identifying* if there exists $\mathbf{W}^Q, \mathbf{W}^K$ such that for any row i and column j,

$$\tilde{\mathbf{P}}_{ij} = \max_k \tilde{\mathbf{P}}_{ik},$$

if, and only, if $\mathbf{A}(G)_{ij} = 1$, where each row can have between one and $(n-1)$ maxima (for connected graphs). Further, an approximation $\mathbf{Q} \in \mathbb{R}^{n\times d}$ to $\mathbf{P}$ is *sufficiently* adjacency-identifying if

$$\big|\big|\mathbf{P} - \mathbf{Q}\big|\big|_{\mathrm{F}} < \epsilon,$$

for any $\epsilon > 0$.

As the name suggests, the property of *(sufficiently) adjacency-identifying* allows us to identify the underlying adjacency matrix during the attention computation. In Section 4, we introduce a slight generalization of the commonly used Laplacian PEs (Kreuzer et al., 2021; Rampášek et al., 2022), which we refer to as LPE, and show that structural embeddings are sufficiently adjacency-identifying when using LPE or SPE (Huang et al., 2024) as node-level PEs. We call a standard transformer using the above tokenization with sufficiently adjacency-identifying node-level PEs the 1-$\mathtt{GT}$.

Let us now understand why the above token embeddings with degree encodings and adjacency-identifying PEs are sufficient to obtain 1-WL expressivity. To this end, we revisit a 1-WL expressive GNN from Grohe (2021), which updates node representations $\mathbf{F}^{(t)}$ at layer t as

$$\mathbf{F}^{(t)} \coloneqq \mathsf{FFN}\big(\mathbf{F}^{(t-1)} + 2\mathbf{A}(G)\mathbf{F}^{(t-1)}\big), \tag{3}$$

where FFN is again a feed-forward neural network and $\mathbf{F}_i^{(t)}$ contains the representation of node $i \in V(G)$ at layer t. Contrast the above to the update of a 1-$\mathtt{GT}$ layer with a single head, given as

$$\mathbf{X}^{(t,1)} \coloneqq \mathsf{FFN}\big(\mathbf{X}^{(t-1,1)} + \mathsf{softmax}\big(\tilde{\mathbf{X}}^{(t-1,1)}\big)\mathbf{X}^{(t-1,1)}\mathbf{W}^{V}\big),$$

where

$$\tilde{\mathbf{X}}^{(t-1,1)} \coloneqq \frac{1}{\sqrt{d_k}}\mathbf{X}^{(t-1,1)}\mathbf{W}^{Q}\big(\mathbf{X}^{(t-1,1)}\mathbf{W}^{K}\big)^{T}$$

denotes the unnormalized attention matrix at layer t. If we now set $\mathbf{W}^{V} = 2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, we obtain

$$\mathbf{X}^{(t,1)} \coloneqq \mathsf{FFN}\big(\mathbf{X}^{(t-1,1)} + 2\cdot\mathsf{softmax}\big(\tilde{\mathbf{X}}^{(t-1,1)}\big)\mathbf{X}^{(t-1,1)}\big).$$

At this point, if we could reconstruct the adjacency matrix with $\mathsf{softmax}(\tilde{\mathbf{X}}^{(t-1,1)})$, we could simulate the 1-WL expressive GNN of Equation (3) with a single attention head. However, the attention matrix is *right-stochastic*, meaning its rows sum to 1. Unfortunately, this is not expressive enough to reconstruct the adjacency matrix, which generally is not right-stochastic. Thus, we aim to reconstruct the *row-normalized adjacency matrix* $\tilde{\mathbf{A}}(G) \coloneqq \mathbf{D}^{-1}\mathbf{A}(G)$, where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix, such that $\mathbf{D}_{ii}$ is the degree of node i. Indeed, $\tilde{\mathbf{A}}(G)$ is right-stochastic. Further, element-wise multiplication of row i of $\tilde{\mathbf{A}}(G)$ with the degree of i recovers $\mathbf{A}(G)$. As we show, adjacency-identifying PEs are, in fact, sufficient for $\mathsf{softmax}(\tilde{\mathbf{X}}^{(t-1,1)})$ to approximate $\tilde{\mathbf{A}}(G)$ arbitrarily close. We show that then, we can use the degree embeddings $\deg(i)$ to de-normalize the i-th row of $\tilde{\mathbf{A}}(G)$ and obtain

$$\mathbf{X}^{(t,1)} = \mathsf{FFN}\big(\mathbf{X}^{(t-1,1)} + 2\mathbf{A}(G)\mathbf{X}^{(t-1,1)}\big).$$

Showing that a transformer can simulate the GNN in Grohe (2021) implies the connection to the 1-WL. We formally prove the above in the following theorem, showing that the 1-$\mathtt{GT}$ can simulate the 1-WL; see Appendix H for proof details.

Theorem 2. *Let $G = (V(G), E(G), \ell)$ be a labeled graph with n nodes and $\mathbf{F} \in \mathbb{R}^{n\times d}$ be a node feature matrix consistent with ℓ. Further, let $C^1_t : V(G) \to \mathbb{N}$ denote the coloring function of the 1-WL at iteration t. Then, for all iterations $t \geq 0$, there exists a parametrization of the 1-$\mathtt{GT}$ such that*

$$C^1_t(v) = C^1_t(w) \iff \mathbf{X}^{(t,1)}(v) = \mathbf{X}^{(t,1)}(w),$$

for all nodes $v, w \in V(G)$.

Having set the stage for theoretically aligned transformers, we now generalize the above to higher orders $k > 1$.

3.2. Transformers with k-WL expressive power

Here, we propose tokenization for a standard transformer, operating on n^k tokens, to be strictly more expressive as the k-WL, for an arbitrary but fixed $k > 1$. Subsequently, to make the architecture more practical, we reduce the number of tokens while remaining strictly more expressive than the 1-WL.

Again, we consider a labeled graph $G = (V(G), E(G), \ell)$ with n nodes and feature matrix $\mathbf{F} \in \mathbb{R}^{n\times d}$, consistent with ℓ. Intuitively, to surpass the limitations of the 1-WL, the k-WL colors ordered subgraphs instead of a single node. More precisely, the k-WL colors the tuples from $V(G)^k$ for $k \geq 2$ instead of the nodes. Of central importance to our tokenization is consistency with the initial coloring of the k-WL and recovering the adjacency information between k-tuples imposed by the k-WL, both of which we will describe hereafter; see Appendix E for a formal definition of the k-WL.

The initial color of a k-tuple $\mathbf{v} \coloneqq (v_1, \ldots, v_k) \in V(G)^k$ under the k-WL depends on its *atomic type* and the labels $\ell(v_1), \ldots, \ell(v_k)$. Intuitively, the atomic type describes the structural dependencies within elements in a tuple. We can represent the atomic type of $\mathbf{v}$ by a $k \times k$ matrix $\mathbf{K}$ over $\{1, 2, 3\}$. That is, the entry $\mathbf{K}_{ij}$ is 1 if $(v_i, v_j) \in E(G)$, 2 if $v_i = v_j$, and 3 otherwise; see Appendix C for a formal definition of the atomic type. Hence, to encode the atomic type as a real vector, we can learn an embedding from the set of $k \times k$ matrices to $\mathbb{R}^e$, where $e > 0$ is the embedding dimension of the atomic type. For a tuple $\mathbf{v}$, we denote this embedding with $\mathrm{atp}(\mathbf{v})$. Apart from the initial colors for tuples, the k-WL also imposes a notion of adjacency between tuples. Concretely, we define

$$\phi_j(\mathbf{v}, w) \coloneqq (v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k),$$

i.e., $\phi_j(\mathbf{v}, w)$ replaces the j-th component of the tuple $\mathbf{v}$ with the vertex w. We say that two tuples are *adjacent* or *j-neighbors* if they are different in the j-th component (or equal, in the case of self-loops).

Now, to construct token embeddings consistent with the initial colors under the k-WL, we initialize n^k token embeddings $\mathbf{X}^{(0,k)} \in \mathbb{R}^{n^k\times d}$, one for each k-tuple. For order k and embedding dimension d of the tuple-level tokens, we first compute node-level tokens $\mathbf{X}^{(0,1)}$ in Equation (1) and, in particular, the structural embeddings $\mathbf{P}(v)$ for each node v, with an embedding dimension of d and then concatenate node-level embeddings along the embedding dimension to construct tuple-level embeddings which are then projected down to fit the embedding dimension d. Specifically, we define the token embedding of a k-tuple $\mathbf{v} = (v_1, \ldots, v_k)$ as

$$\mathbf{X}^{(0,k)}(\mathbf{v}) \coloneqq \big[\mathbf{X}^{(0,1)}(v_i)\big]_{i=1}^{k}\mathbf{W} + \mathrm{atp}(\mathbf{v}), \qquad (4)$$

where $\mathbf{W} \in \mathbb{R}^{d\cdot k\times d}$ is a projection matrix. Intuitively, the above construction ensures that the token embeddings respect the initial node colors and the atomic type. Analogously to

our notion of adjacency-identifying, we use the structural embeddings $\mathbf{P}(v_i)$ in each node embedding $\mathbf{X}^{(0,1)}(v_i)$ to identify the j-neighborhood adjacency between tuples $\mathbf{v}$ and $\mathbf{w}$ in the j-th attention head. To reconstruct the j-neighborhood adjacency, we show that it is sufficient to identify the nodes in each tuple-level token. Hence, we define the following requirements for the structural embeddings $\mathbf{P}$.

Definition 3 (Node-identifying)**.** Let $G = (V(G), E(G), \ell)$ be a graph with n nodes. Let $\mathbf{X}^{(0,k)} \in \mathbb{R}^{n^k \times d}$ denote the initial token embeddings according to Equation (4) with structural embeddings $\mathbf{P} \in \mathbb{R}^{n \times d}$. Let further

$$\tilde{\mathbf{P}} \coloneqq \frac{1}{\sqrt{d_k}} \mathbf{P}\mathbf{W}^Q \big(\mathbf{P}\mathbf{W}^K\big)^T,$$

where $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d \times d}$. Then $\mathbf{P}$ is *node-identifying* if there exists $\mathbf{W}^Q, \mathbf{W}^K$ such that for any row i and column j,

$$\tilde{\mathbf{P}}_{ij} = \max_k \tilde{\mathbf{P}}_{ik},$$

if, and only, if $i = j$. Further, an approximation $\mathbf{Q} \in \mathbb{R}^{n \times d}$ of $\mathbf{P}$ is sufficiently node-identifying if

$$\big|\big|\mathbf{P} - \mathbf{Q}\big|\big|_{\mathrm{F}} < \epsilon,$$

for any $\epsilon > 0$.

As we will show, the above requirement allows the attention to distinguish whether two tuples share the same nodes, intuitively, by counting the number of node-to-node matches between two tuples, which is sufficient to determine whether the tuples are j-neighbors. For structural embeddings $\mathbf{P}$ to be node-identifying, it suffices, for example, that $\mathbf{P}$ has an orthogonal sub-matrix.

In Section 4, we show that structural embeddings are also sufficiently node-identifying when using LPE or SPE (Huang et al., 2024) as node-level PEs. It should also be mentioned that our definition of node-identifying structural embeddings generalizes the node identifiers in Kim et al. (2022), i.e., their node identifiers are node-identifying. Still, structural embeddings exist that are (sufficiently) node-identifying and do not qualify as node identifiers in Kim et al. (2022).

We call a standard transformer using the above tokenization with sufficiently node- and adjacency-identifying structural embeddings the k-GT. We then show the connection of the k-GT to the k-WL, resulting in the following theorem; see Appendix I for proof details.

Theorem 4. *Let $G = (V(G), E(G), \ell)$ be a labeled graph with n nodes and $k \geq 2$ and $\mathbf{F} \in \mathbb{R}^{n \times d}$ be a node feature matrix consistent with ℓ. Let C^k_t denote the coloring function of the k-WL at iteration t. Then for all iterations $t \geq 0$, there exists a parametrization of the k-GT such that*

$$C^k_t(\mathbf{v}) = C^k_t(\mathbf{w}) \iff \mathbf{X}^{(t,k)}(\mathbf{v}) = \mathbf{X}^{(t,k)}(\mathbf{w})$$

for all k-tuples $\mathbf{v}$ and $\mathbf{w} \in V(G)^k$.

Note that similar to Theorem 2, the above theorem gives a lower bound on the expressivity of the k-GT. We now show that the k-GT is strictly more powerful than the k-WL by showing that the k-GT can also simulate the δ-k-WL, a k-WL variant that, for each j-neighbor $\phi_j(\mathbf{v}, w)$, additionally considers whether $(v_j, w) \in E(G)$ (Morris et al., 2020). That is, the δ-k-WL distinguishes for each j-neighbor whether the replaced node and the replacing node share an edge in G. Morris et al. (2020) showed that the δ-k-WL is strictly more expressive than the k-WL. We use this to show that the k-GT is strictly more expressive than the k-WL, implying the following result; see again Appendix I for proof details.

Theorem 5. *For $k > 1$, the k-GT is strictly more expressive than the k-WL.*

Note that the k-GT uses n^k tokens. However, for larger k and large graphs, the number of tokens might present an obstacle in practice. Luckily, our close alignment of the k-GT with the k-WL hierarchy allows us to directly benefit from a recent result in Morris et al. (2022), reducing the number of tokens while maintaining an expressivity guarantee. Specifically, Morris et al. (2022) define the set of (k, s)-tuples $V(G)^k_s$ as

$$V(G)^k_s \coloneqq \{\mathbf{v} \in V(G)^k \mid \#\mathrm{comp}(G[\mathbf{v}]) \leq s\},$$

where $\#\mathrm{comp}(H)$ denotes the number of connected components in subgraph H, $G[\mathbf{v}]$ denotes the ordered subgraph of G, induced by the nodes in $\mathbf{v}$ and $s \leq k$ is a hyper-parameter. Morris et al. (2022) then define the (k, s)-LWL as a k-WL variant that only considers the tuples in $V(G)^k_s$ and additional only use subset of adjacent tuples; see (Morris et al., 2022) and Appendix E for a detailed description of the (k, s)-WL. Fortunately, the runtime of the (k, s)-LWL depends only on k, s, and the sparsity of the graph, resulting in a more efficient and scalable k-WL variant. We can now directly translate this modification to our token embeddings. Specifically, we call the k-GT using only the token embeddings $\mathbf{X}^{(0,k)}(\mathbf{v})$ where $\mathbf{v} \in V(G)^k_s$, the (k, s)-GT.

Now, Morris et al. (2022, Theorem 1) shows that, for $k > 1$, the $(k + 1, 1)$-LWL is strictly stronger than the $(k, 1)$-LWL. Further, note that the $(1, 1)$-LWL is equal to the 1-WL. Using this result, we can prove the following; see Appendix I for proof details.

Theorem 6. *For all $k > 1$, the $(k, 1)$-GT is strictly more expressive than the $(k - 1, 1)$-GT. Further, the $(2, 1)$-GT is strictly more expressive than the 1-WL.*

Note that the $(2, 1)$-GT only requires $\mathcal{O}(n + m)$ tokens, where m is the number of edges, and consequently has a runtime complexity of $\mathcal{O}((n + m)^2)$, the same as TokenGT. Hence, the $(2, 1)$-GT improves the expressivity result of TokenGT, which is shown to have 1-WL expressive power (Kim et al., 2022) while having the same runtime complexity.

In summary, our alignment of standard transformers with

Table 1: Comparison of our theoretical results with Kim et al. (2021) and Kim et al. (2022). Highlighted are aspects in which our results improve over Kim et al. (2022). Here, n denotes the number of nodes, and m denotes the number of edges. We denote with $\sqsubset A$ that the transformer is strictly more expressive than algorithm A. Note that for pure transformers, the squared complexity can be relaxed to linear complexity when applying linear attention approximation such as the Performer (Choromanski et al., 2021).

Transformer	Provable expressivity	Runtime complexity	# Heads	Pure transformer
Sparse Transformer (Kim et al., 2021)	$\sqsubset$ Gilmer et al. (2017)	$\mathcal{O}(m)$	1	✗
TokenGT (Kim et al., 2022)	1-WL	$\mathcal{O}((n+m)^2)$	15	✓
1-GT (ours)	1-WL	$\mathcal{O}(n^2)$	1	✓
$(2,1)$-GT (ours)	$\sqsubset$ 1-WL	$\mathcal{O}((n+m)^2)$	2	✓
Higher-Order Transformer (Kim et al., 2021)	k-WL	$\mathcal{O}(n^{2k})$	1	✗
k-order Transformer (Kim et al., 2022)	k-WL	$\mathcal{O}(n^{2k})$	*bell*(2k)	✓
k-GT (ours)	$\sqsubset$ k-WL	$\mathcal{O}(n^{2k})$	$2k$	✓

the k-WL hierarchy has brought forth a variety of improved theoretical results for pure transformers, providing a strict increase in provable expressive power for every order $k > 0$ compared to previous works. See Table 1 for a direct comparison of our results with Kim et al. (2021) and Kim et al. (2022). Most importantly, Theorem 6 leads to a hierarchy of increasingly expressive pure transformers, taking one step closer to answering our guiding question in Section 1. In what follows, we make our transformers feasible in practice.

4. Implementation details

Here, we discuss the implementation details of pure transformers. In particular, we demonstrate that Laplacian PEs (Kreuzer et al., 2021) are sufficient to be used as node-level PEs for our transformers. Afterward, we introduce *order transfer*, a way to transfer model weights between different orders of the WL hierarchy; see Appendix A for additional implementation details.

4.1. Node and adjacency-identifying PEs

Here, we develop PEs based on Laplacian eigenvectors and -values that are both sufficiently node- *and* adjacency-identifying, avoiding computing separate PEs for node- and adjacency identification; see Appendix A.1 for details about the graph Laplacian and some background on PEs based on the spectrum of the graph Laplacian.

Let $\lambda \coloneqq (\lambda_1, \ldots, \lambda_l)^T$ denote the vector of the l smallest, possibly repeated, eigenvalues of the graph Laplacian for some graph with n nodes and let $\mathbf{V} \in \mathbb{R}^{n \times l}$ be a matrix such that the i-th column of $\mathbf{V}$ is the eigenvector corresponding to eigenvalue λ_i. We will now briefly introduce LPE, based on Kreuzer et al. (2021), as well as SPE from Huang et al. (2024) and show that these node-level PEs are node- *and* adjacency-identifying.

LPE Here, we define a slight generalization of the Laplacian PEs in Kreuzer et al. (2021) that enables sufficiently node and adjacency-identifying PEs. Concretely, we define LPE as

$$\text{LPE}(\mathbf{V}, \lambda) = \rho\Big(\big[\phi(\mathbf{V}_1^T, \lambda + \epsilon) \quad \ldots \quad \phi(\mathbf{V}_n^T, \lambda + \epsilon)\big]\Big), \quad (5)$$

where $\epsilon \in \mathbb{R}^l$ is a learnable, zero-initialized vector, we introduce to show our result. Here, $\phi\colon \mathbb{R}^2 \to \mathbb{R}^d$ is an FFN that is applied row-wise and $\rho\colon \mathbb{R}^{l \times d} \to \mathbb{R}^d$ is a permutation-equivariant neural network, applied row-wise. One can recover the Laplacian PEs in Kreuzer et al. (2021) by setting $\epsilon = \mathbf{0}$ and implementing ρ as first performing a sum over the first dimension of its input, resulting in an d-dimensional vector and then subsequently applying an FFN to obtain a d-dimensional output vector. Note that then, Equation (5) forms a DeepSet (Zaheer et al., 2017) over the set of eigenvector components, paired with their (ϵ-perturbed) eigenvalues. While slightly deviating from the Laplacian PEs defined in Kreuzer et al. (2021), the ϵ vector is necessary to ensure that no spectral information is lost when applying ρ.

We will now show that LPE is sufficiently node- and adjacency-identifying. Further, this result holds irrespective of whether the Laplacian is normalized. As a result, LPE can be used with 1-GT, k-GT, and (k,s)-GT; see Appendix F.3 for proof details.

Theorem 7. *Structural embeddings with LPE as node-level PE are sufficiently node- and adjacency-identifying.*

SPE We also show that SPE (Huang et al., 2024) are sufficiently node- and adjacency-identifying. Huang et al. (2024) define the SPE encodings as

$$\text{SPE}(\mathbf{V}, \lambda) = \rho\Big(\big[\mathbf{V}\text{diag}(\phi_1(\lambda))\mathbf{V}^T \quad \ldots \quad \mathbf{V}\text{diag}(\phi_n(\lambda))\mathbf{V}^T\big]\Big),$$

where $\phi_1, \ldots, \phi_n\colon \mathbb{R}^l \to \mathbb{R}^l$ are equivariant FFNs and $\rho\colon \mathbb{R}^{n \times l} \to \mathbb{R}^d$ is a permutation equivariant neural network,

applied row-wise. For example, ρ can be implemented by first performing a sum over the first dimension of its input, resulting in an l-dimensional vector, and then subsequently applying an FFN to obtain a d-dimensional output vector.

SPE have the benefit of being *stable*, meaning a small perturbation in the graph structure causes only a small change in the resulting SPE encoding. As a result, stability can be related to OOD generalization (Huang et al., 2024). We can show that SPE are sufficiently node- and adjacency-identifying. As a result, SPE encodings can be used with 1-GT, k-GT and (k, s)-GT; see Appendix F.4 for proof details.

Theorem 8. *Structural embeddings with SPE as node-level PE are sufficiently node- and adjacency-identifying.*

4.2. Order transfer

Here, we describe order transfer, a strategy for effectively using the k-GT for $k > 2$ in practice. The idea of order transfer consists of pre-training a transformer on a lower-order tokenization, such as on the $(2, 1)$-GT tokens, and subsequently fine-tuning the pre-trained weights on a higher-order transformer, such as the 2-GT, 3-GT or the $(3, 1)$-GT. We evaluate order transfer empirically in Section 5.2.

Order transfer could be useful when pre-training a higher-order model from scratch is infeasible. However, using a higher-order model on a smaller downstream task is feasible and beneficial to performance. Examples of such scenarios are (a) a large difference in the number of graphs in the pre-training dataset compared to the fine-tuning task, (b) a large difference in the size of the graphs in the pre-training dataset compared to the fine-tuning task, or (c) a specific fine-tuning task benefiting strongly from higher-order expressivity. As a result, order transfer is a promising application of expensive yet expressive higher-order models.

5. Experimental evaluation

Here, we conduct an experimental evaluation of our theoretical results. In particular, we consider two settings. To motivate the use of large-scale pure transformers for graph learning, we evaluate the empirical performance of our transformers under a pre-training/fine-tuning paradigm typically employed with large-scale language, vision, and graph models (Devlin et al., 2019; Dosovitskiy et al., 2021; Ying et al., 2021). This approach is especially promising in molecular learning, where downstream datasets often only contain a few thousand examples (Hu et al., 2020b; Méndez-Lucio et al., 2022); see Section 5.1 for pre-training and Section 5.2 for fine-tuning. Moreover, we benchmark the $(2, 1)$-GT and $(3, 1)$-GT on how well these models can leverage their expressivity in practice; see Section 5.3. The source code for all experiments is available at https://github.com/luis-mueller/wl-transformers.

5.1. Pre-training

For pre-training, we train on PCQM4MV2, one of the largest molecular regression datasets available (Hu et al., 2021). The dataset consists of roughly 3.8M molecules, and the task is to predict the HOMO-LUMO energy gap. Here, we pre-train the $(2, 1)$-GT for 2M steps with a cosine learning rate schedule with 60K warm-up steps on two A100 NVIDIA GPUs; see Table 6 in Appendix B for details on the hyper-parameters. For model evaluation, we use the code provided by Hu et al. (2021), available at https://github.com/snap-stanford/ogb. Further, to demonstrate the effectiveness of our transformers on large node-level tasks, we additionally benchmark the $(2, 1)$-GT on CS and PHOTO, two transductive node classification datasets (Shchur et al., 2018). On all three tasks, we compare to Graphormer (Ying et al., 2021), a strong graph transformer with modified attention, and TokenGT (Kim et al., 2022) as a pure transformer baseline. We use 12 transformer layers, a hidden dimension of 768, 16 attention heads, and a GELU non-linearity (Hendrycks & Gimpel, 2016). As node-level PEs, we compare both LPE and SPE, as defined in Section 4.

We present our results in Table 2 and observe that the $(2, 1)$-GT further closes the gap between pure transformers and graph transformers with graph inductive bias such as Graphormer (Ying et al., 2021). Considering that the $(2, 1)$-GT has still relatively low graph inductive bias and in light of recent advances in terms of quality and size of pre-training dataset (Beaini et al., 2024), we expect this gap to further shrink with increasing data scale. Further, we find that on all three datasets, LPE are favorable or comparable to SPE.

5.2. Fine-tuning

Here, we describe our fine-tuning experiments. When fine-tuning, we re-use the pre-trained weights for the transformer layers and randomly initialize new weights for task-specific feature encodings and the task head. To demonstrate that fine-tuning large pre-trained transformers for small downstream tasks is feasible even with a smaller compute budget, we ensure that all experiments can be run on a single A10 Nvidia GPU with 24GB RAM.

Molecular regression To determine whether pre-training can improve the fine-tuning performance of our transformers, we choose ALCHEMY (12K), a small-scale molecular dataset with 12K molecules (Morris et al., 2022). Specifically, we evaluate three settings: (a) training the $(2, 1)$-GT from scratch, (b) fine-tuning the pre-trained $(2, 1)$-GT and (c) order transfer from $(2, 1)$-GT to $(3, 1)$-GT, where we re-use the pre-trained weights from the $(2, 1)$-GT pre-training but learn a new tokenizer with $(3, 1)$-GT tokens. We evaluate these settings both for LPE and SPE as node-level PEs. We compare the results to three GNNs with node-level PEs

Table 2: Comparison of $(2,1)$-GT to Graphormer and TokenGT. Results on PCQM4Mv2 over a single random seed, as well as CS and PHOTO over 7 random seeds where we also report standard deviation. Results for baselines are taken from Kim et al. (2022). We highlight **best** and **second** best model on each dataset.

Model	PCQM4Mv2 Validation MAE ↓	CS Accuracy ↑	PHOTO Accuracy ↑
Graphormer	**0.0864**	0.791 ± 0.015	0.894 ± 0.004
TokenGT	0.0910	0.903 ± 0.004	**0.949** ± 0.007
$(2,1)$-GT + LPE	**0.0870**	**0.924** ± 0.008	0.933 ± 0.013
$(2,1)$-GT + SPE	0.0888	**0.920** ± 0.002	**0.933** ± 0.011

based on the eigenvectors and -values of the graph Laplacian: SignNet, BasisNet, and SPE (Huang et al., 2024); see Table 3 for results. Here, we find that even without pre-training, the $(2,1)$-GT already performs well on ALCHEMY. Most notably, the $(2,1)$-GT with SPE without fine-tuning already performs on par with SignNet. Nonetheless, we observe significant improvements through pre-training $(2,1)$-GT. Most notably, fine-tuning the pre-trained $(2,1)$-GT with LPE or SPE beats all GNN baselines. Interestingly, training the $(2,1)$-GT with SPE from scratch results in much better performance than training the $(2,1)$-GT with LPE from scratch. However, once pre-trained, the $(2,1)$-GT performs better with LPE than with SPE. Moreover, we observe no improvements when performing order transfer to the $(3,1)$-GT. However, the $(3,1)$-GT with LPE and pre-trained weights from the $(2,1)$-GT beats SignNet (Lim et al., 2023), a strong GNN baseline. Further, $(3,1)$-GT with SPE performs on par with SPE in Huang et al. (2024), the best of our GNN baselines. Interestingly, order transfer improves over the $(2,1)$-GT trained from scratch for both LPE and SPE. As a result, we hypothesize that LPE and SPE provide sufficient expressivity for this task but that pre-training is required to fully leverage their potential. Further, we hypothesize that the added $(3,1)$-GT tokens lead to overfitting.

We conclude that pre-training can help our transformers' downstream performance. Further, order transfer is a promising technique for fine-tuning downstream tasks, particularly those that benefit from higher-order representations. However, its benefits might be nullified in the presence of sufficiently expressive node-level PEs in combination with large-scale pre-training.

Molecular classification Here, we evaluate whether fine-tuning a large pre-trained transformer can be competitive with a strong GNN baseline on five small-scale molecular classification tasks, namely BBBP, BACE, CLINTOX, TOX21, and TOXCAST (Hu et al., 2020a). The number of molecules in any of these datasets is lower than 10K, making them an ideal choice to benchmark whether large-scale pre-trained models such as the $(2,1)$-GT can compete with a task-specific GNN. Specifically, following Tönshoff et al. (2023), we aim for a fair comparison by carefully designing and hyper-parameter tuning a GINE model (Xu et al., 2019b) with residual connections, batch normalization, dropout, GELU non-linearities and, most crucially, the same node-level PEs as the $(2,1)$-GT. We followed Hu et al. (2020b) in choosing the GINE layer over other GNN layers due to its guaranteed 1-WL expressivity. It is worth noting that our GINE with LPE encodings significantly outperforms the GIN in Hu et al. (2020b) without pre-training on all five datasets, demonstrating the quality of this baseline. Interestingly, GINE with SPE as node-level PEs underperforms against the GIN in Hu et al. (2020b) on all but one datasets. In addition, we report the best pre-trained model for each task in Hu et al. (2020b). We fine-tune the $(2,1)$-GT for 5 epochs and train the GINE model for 45 epochs. In addition, we also train the $(2,1)$-GT for 45 epochs from scratch to study the impact of pre-training on the molecular classification tasks. Note that we do not pre-train our GINE model to evaluate whether a large pre-trained transformer can beat a small GNN model trained from scratch; see Table 4 for results.

Table 3: Effect of pre-training for fine-tuning performance on ALCHEMY (12K). Pre-trained weights for both $(2,1)$-GT and $(3,1)$-GT are taken from the $(2,1)$-GT model in Table 2. We report mean and standard deviation over 3 random seeds. Baseline results for SignNet, BasisNet and SPE are taken from Huang et al. (2024). We highlight the **best**, **second** best and **third** best model.

Model	Pre-trained	ALCHEMY (12K) MAE ↓
SignNet	✗	0.113 ± 0.002
BasisNet	✗	0.110 ± 0.001
SPE	✗	**0.108** ± 0.001
$(2,1)$-GT + LPE	✗	0.124 ± 0.001
	✓	**0.101** ± 0.001
$(3,1)$-GT + LPE	✓	0.114 ± 0.001
$(2,1)$-GT + SPE	✗	0.112 ± 0.000
	✓	**0.103** ± 0.002
$(3,1)$-GT + SPE	✓	**0.108** ± 0.001

First, we find that the pre-trained $(2,1)$-GT with SPE is better than the corresponding GINE with SPE on all five datasets. The pre-trained $(2,1)$-GT with LPE is better than GINE with LPE on all but one dataset. Moreover, the $(2,1)$-GT with LPE as well as the $(2,1)$-GT with SPE are each better or on par with the best pre-trained GIN in Hu et al. (2020b) in two out of five datasets. When studying the impact of pre-training, we observe that pre-training leads mostly to performance improvement. The notable exception is on the

Table 4: Fine-tuning results on small OGB molecular datasets compared to a fully-equipped and -tuned GIN. Results over 6 random seeds. We highlight **best**, **second** best and **third** best model for each dataset. Ties are broken by smaller standard deviation. For comparison, we also report pre-training results from Hu et al. (2020b).

Model	Pre-trained	BBBP	BACE	ClinTox	Tox21	ToxCast
		AUROC ↑	AUROC ↑	AUROC ↑	AUROC ↑	AUROC ↑
GIN (Hu et al., 2020b)	✗	0.658 ± 0.045	0.701 ± 0.054	0.580 ± 0.044	0.740 ± 0.008	0.634 ± 0.006
	✓	0.688 ± 0.008	**0.845 ± 0.007**	0.737 ± 0.028	**0.783 ± 0.003**	**0.665 ± 0.003**
GINE+LPE	✗	0.670 ± 0.016	0.763 ± 0.007	0.824 ± 0.016	0.763 ± 0.011	**0.659 ± 0.005**
GINE+SPE	✗	0.613 ± 0.036	0.677 ± 0.047	0.641 ± 0.066	0.709 ± 0.023	0.593 ± 0.018
$(2,1)$-GT +LPE	✗	**0.703± 0.025**	0.786± 0.019	0.821± 0.045	**0.763± 0.003**	**0.667± 0.007**
$(2,1)$-GT +LPE	✓	0.679± 0.012	**0.815± 0.017**	**0.867± 0.020**	**0.780± 0.005**	0.642± 0.007
$(2,1)$-GT +SPE	✗	**0.694± 0.020**	0.776± 0.030	**0.845± 0.017**	0.758± 0.006	0.657± 0.007
$(2,1)$-GT +SPE	✓	**0.703± 0.013**	**0.793± 0.025**	**0.859± 0.021**	0.762± 0.003	0.637± 0.003

ToxCast dataset, where the pre-trained $(2,1)$-GT underperforms even the GIN baselines without pre-training. Hence, we conclude that the pre-training/fine-tuning paradigm is viable for applying pure transformers such as the $(2,1)$-GT to small-scale problems.

5.3. Expressivity tests

Since we only provide expressivity lower bounds, we empirically investigate the expressive power of the k-GT. To this end, we evaluate the BREC benchmark offering fine-grained and scalable expressivity tests (Wang & Zhang, 2023). The benchmark is comprised of 400 graph pairs that range from being 1-WL to 4-WL indistinguishable and pose a challenge even to the most expressive models; see Table 5 for the expressivity results of $(2,1)$-GT and $(3,1)$-GT on BREC. We mainly compare to our GINE baseline, Graphormer (Ying et al., 2021) as a graph transformer baseline, and the 3-WL as a potential expressivity upper-bound. In addition, we compare our GINE model, $(2,1)$-GT and $(3,1)$-GT for both LPE and SPE. We find that SPE encodings consistently lead to better performance. Further, we observe that increased expressivity also leads to better performance, as for both LPE and SPE, the $(2,1)$-GT beats our GINE baseline, and the $(3,1)$-GT beats the $(2,1)$-GT across all tasks. Finally, we find that both the $(2,1)$-GT and the $(3,1)$-GT improve over Graphormer while still being outperformed by the 3-WL, irrespective of the choice of PE.

Table 5: Results on the BREC benchmark over a single seed. Baseline results are taken from Wang & Zhang (2023). We highlight the best model for each PE and category (excluding the 3-WL, which is not a model and merely serves as a reference point). We additionally report the results of Graphormer from Wang & Zhang (2023).

Model	PE	Basic	Regular	Extension	CFI	*All*
GINE		19	20	37	**3**	79
$(2,1)$-GT	LPE	36	28	51	**3**	118
$(3,1)$-GT		**55**	**46**	**55**	**3**	**159**
GINE		56	48	93	20	217
$(2,1)$-GT	SPE	**60**	**50**	**98**	18	226
$(3,1)$-GT		**60**	**50**	97	**21**	**228**
Graphormer		16	12	41	10	79
3-WL		60	50	100	60	270

6. Conclusion

In this work, we propose a hierarchy of expressive pure transformers that are also feasible in practice. We improve existing pure transformers such as TokenGT (Kim et al., 2022) in several aspects, both theoretically and empirically. Theoretically, our hierarchy has stronger provable expressivity for each k and is more scalable. For example, our $(2,1)$-GT and $(3,1)$-GT have a provable expressivity strictly above 1-WL but are also feasible in practice. Empirically, we verify our claims about practical feasibility and show that our transformers improve over existing pure transformers, closing the gap between pure transformers and graph transformers with strong graph inductive bias on the large-scale PCQM4Mv2 dataset. Further, we show that fine-tuning our pre-trained transformers is a feasible and resource-efficient approach for applying pure transformers to small-scale datasets. We discover that higher-order transformers can efficiently re-use pre-trained weights from lower-order transformers during fine-tuning, indicating a promising direction for utilizing higher-order expressivity to improve results on downstream tasks. Future work could explore aligning transformers to other variants of Weisfeiler–Leman. Finally, pre-training pure transformers on truly large-scale datasets such as those recently proposed by Beaini et al. (2024) could be a promising direction for graph transformers.

Impact statement

This paper presents work whose goal is to advance the field of machine learning. Our work has many potential societal consequences, none of which must be specifically highlighted here.

Acknowledgements

CM and LM are partially funded by a DFG Emmy Noether grant (468502433) and RWTH Junior Principal Investigator Fellowship under Germany's Excellence Strategy.

References

Anderson, W. N. and Morley, T. D. Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra*, 18 (2):141–145, 1985.

Azizian, W. and Lelarge, M. Characterizing the expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2021.

Babai, L. Lectures on graph isomorphism. University of Toronto, Department of Computer Science. Mimeographed lecture notes, 1979.

Beaini, D., Huang, S., Cunha, J. A., Li, Z., Moisescu-Pareja, G., Dymov, O., Maddrell-Mander, S., McLean, C., Wenkel, F., Müller, L., Mohamud, J. H., Parviz, A., Craig, M., Koziarski, M., Lu, J., Zhu, Z., Gabellini, C., Klaser, K., Dean, J., Wognum, C., Sypetkowski, M., Rabusseau, G., Rabbany, R., Tang, J., Morris, C., Koutis, I., Ravanelli, M., Wolf, G., Tossou, P., Mary, H., Bois, T., Fitzgibbon, A. W., Banaszewski, B., Martin, C., and Masters, D. Towards foundational models for molecular learning on large-scale multi-task datasets. In *International Conference on Learning Representations*, 2024.

Bodnar, C., Frasca, F., Otter, N., Wang, Y. G., Liò, P., Montúfar, G., and Bronstein, M. M. Weisfeiler and Lehman go cellular: CW networks. In *Advances in Neural Information Processing Systems*, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Cai, J., Fürer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, 2019.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Geerts, F. and Reutter, J. L. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

Glickman, D. and Yahav, E. Diffusing graph attention. *ArXiv preprint*, 2023.

Grohe, M. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

Grohe, M. The logic of graph neural networks. In *Symposium on Logic in Computer Science*, 2021.

He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning*, 2023.

Hendrycks, D. and Gimpel, K. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv preprint*, 2016.

Horn, R. A. and Johnson, C. R. *Matrix Analysis, 2nd Edition*. Cambridge University Press, 2012.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V. S., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. OGB-LSC: A large-scale challenge for machine learning on graphs. In *NeurIPS: Datasets and Benchmarks Track*, 2021.

Huang, Y., Lu, W., Robinson, J., Yang, Y., Zhang, M., Jegelka, S., and Li, P. On the stability of expressive positional encodings for graphs. In *International Conference on Learning Representations*, 2024.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, 2021.

Kim, J., Oh, S., and Hong, S. Transformers generalize deepsets and can be extended to graphs & hypergraphs. In *Advances in Neural Information Processing Systems*, 2021.

Kim, J., Nguyen, D., Min, S., Cho, S., Lee, M., Lee, H., and Hong, S. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kreuzer, D., Beaini, D., Hamilton, W. L., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems*, 2021.

Lim, D., Robinson, J. D., Zhao, L., Smidt, T. E., Sra, S., Maron, H., and Jegelka, S. Sign and basis invariant networks for spectral graph representation learning. In *International Conference on Learning Representations*, 2023.

Lipman, Y., Puny, O., and Ben-Hamu, H. Global attention improves graph networks generalization. *ArXiv preprint*, 2020.

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, K., Coates, M., H.S. Torr, P., and Lim, S.-N. Graph Inductive Biases in Transformers without Message Passing. In *International Conference on Machine Learning*, 2023.

Malkin, P. N. Sherali–Adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 2014.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, 2019a.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019b.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In *International Conference on Machine Learning*, 2019c.

Méndez-Lucio, O., Nicolaou, C. A., and Earnshaw, B. Mole: a molecular foundation model for drug discovery. *ArXiv preprint*, 2022.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2019.

Morris, C., Rattan, G., and Mutzel, P. Weisfeiler and Leman go sparse: Towards higher-order graph embeddings. In *Advances in Neural Information Processing Systems*, 2020.

Morris, C., Rattan, G., Kiefer, S., and Ravanbakhsh, S. SpeqNets: Sparsity-aware permutation-equivariant graph networks. In *International Conference on Machine Learning*, 2022.

Morris, C., L., Y., Maron, H., Rieck, B., Kriege, N. M., Grohe, M., Fey, M., and Borgwardt, K. Weisfeiler and Leman go machine learning: The story so far. *Journal of Machine Learning Research*, 2023.

Müller, L., Galkin, M., Morris, C., and Rampásek, L. Attending to graph transformers. *Transactions on Machine Learning Research*, 2024.

Puny, O., Lim, D., Kiani, B. T., Maron, H., and Lipman, Y. Equivariant polynomials for graph neural networks. In *International Conference on Machine Learning*, 2023.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 2022.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *ArXiv preprint*, 2018.

Tönshoff, J., Ritzert, M., Rosenbluth, E., and Grohe, M. Where did the gap go? reassessing the long-range graph benchmark. *ArXiv preprint*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Wang, Y. and Zhang, M. Towards better evaluation of GNN expressiveness with BREC dataset. *ArXiv preprint*, 2023.

Weisfeiler, B. and Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019a.

Xu, K., Wang, L., Yu, M., Feng, Y., Song, Y., Wang, Z., and Yu, D. Cross-lingual knowledge graph alignment via graph matching neural network. In *Annual Meeting of the Association for Computational Linguistics*, 2019b.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing System*, 2021.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems*, 2017.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the expressive power of gnns via graph biconnectivity. In *International Conference on Learning Representations*, 2023.

A. Additional implementation details

Here, we outline additional implementation details that make our pure transformers more feasible in practice.

A.1. Positional encodings based on Laplacian eigenmaps

Encoding Laplacian eigenmaps are a popular choice for PEs in GNNs and graph transformers (Kreuzer et al., 2021; Lim et al., 2023; Müller et al., 2024; Rampášek et al., 2022). In particular, Lim et al. (2023) show that their PEs based on Laplacian eigenmaps generalize other PEs, such as those based on random walks or heat kernels. In what follows, we will briefly review some background on Laplacian eigenvectors and -values as input to machine learning models and then show how such PEs can be constructed to be node- and adjacency-identifying.

For an n-order graph G, the *graph Laplacian* is defined as $\mathbf{L} \coloneqq \mathbf{D} - \mathbf{A}(G)$, where $\mathbf{D}$ denotes the degree matrix and $\mathbf{A}(G)$ denotes the adjacency matrix. We then consider the eigendecomposition of $\mathbf{L} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$, the column vectors of $\mathbf{V}$ correspond to eigenvectors and the diagonal matrix $\mathbf{\Sigma}$ contains the corresponding eigenvalues of the graph Laplacian. Since $\mathbf{L}$ is real and symmetric, such decomposition always exists; see, e.g., Corollary 2.5.11 in Horn & Johnson (2012). In addition, the eigenvalues of $\mathbf{L}$ are real and non-negative. Some works also consider the *normalized graph Laplacian* $\tilde{\mathbf{L}} \coloneqq D^{-\frac{1}{2}}\mathbf{L}D^{-\frac{1}{2}}$ (Kreuzer et al., 2021; Lim et al., 2023). The statements we made for the graph Laplacian also hold for the normalized graph Laplacian, i.e., $\tilde{\mathbf{L}}$ is real and symmetric, and its eigenvalues are real and non-negative. Further, the eigenvectors of real and symmetric matrices are orthonormal (Lim et al., 2023).

We need to address the following to use Laplacian eigenvectors for machine learning. Let $\mathbf{v}$ be an eigenvector of a matrix $\mathbf{M}$ with eigenvalue λ. Then, $-\mathbf{v}$ is also an eigenvector of $\mathbf{M}$ with eigenvalue λ. As a result, we want a machine learning model to be invariant to the signs of the eigenvectors. Kreuzer et al. (2021) address this issue by randomly flipping the sign of each eigenvector during training for the model to learn this invariance from data. Concretely, given k eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_k$, we uniformly sample k independent sign values $s_1, \dots, s_k \in \{-1, 1\}$ and then plug eigenvectors $s_1 \cdot \mathbf{v}_1, \dots, s_k \cdot \mathbf{v}_k$ into the model during training.

A.2. Implementation of LPE and SPE

Here, we describe our concrete implementations of LPE and SPE based on our definition in Section 4.

For LPE, we implement ρ by performing a sum over its input's first dimension and applying an FFN, as described in Section 4. We choose this implementation to stay as close to implementing the Laplacian PEs in Kreuzer et al. (2021).

For SPE, we follow the implementation in Huang et al. (2024). Concretely, let

$$\mathbf{Q} \coloneqq \begin{bmatrix} \mathbf{V}\mathrm{diag}(\phi_1(\lambda))\mathbf{V}^T & \dots & \mathbf{V}\mathrm{diag}(\phi_n(\lambda))\mathbf{V}^T \end{bmatrix} \in \mathbb{R}^{n \times n \times l}$$

be the input tensor to ρ. Huang et al. (2024) propose to partition $\mathbf{Q}$ along the second axis into n matrices of size $n \times l$ and then to apply a GIN to each of those n matrices in parallel where we use the adjacency matrix of the original graph. Concretely, the GIN maps each $n \times l$ matrix to a matrix of shape $n \times d$, where d is the output dimension of ρ. Finally, the output of ρ is the sum of all $n \times d$ matrices. We adopt this implementation for our experiments. For prohibitively large graphs, we propose the following modification to the above implementation of ρ. Specifically, we let $\mathbf{V}_{:m}$ denote the matrix containing as columns the eigenvectors corresponding to the m smallest eigenvalues. Then, we define

$$\mathbf{Q}_{:m} \coloneqq \begin{bmatrix} \mathbf{V}\mathrm{diag}(\phi_1(\lambda))\mathbf{V}_{:m}^T & \dots & \mathbf{V}\mathrm{diag}(\phi_n(\lambda))\mathbf{V}_{:m}^T \end{bmatrix} \in \mathbb{R}^{n \times m \times l} \tag{6}$$

and use the same ρ as for the case $m = n$.

A.3. Atomic types from edge embeddings

Here, we discuss a practical implementation of the atomic type embeddings atp in Section 3.2. In particular, we use the fact that most graph learning datasets also include edge types, for example, the bond type in molecular datasets. Now, for an undirected graph G and a tuple $\mathbf{v} = (v_1, \dots, v_k) \in V(G)^k_s$, let $\mathbf{E}(v_i, v_j) \in \mathbb{R}^d$ denote a learnable embedding of the edge type between nodes v_i and v_j, where we additionally set $\mathbf{E}(v_i, v_j) = \mathbf{0}$ if, and only, if v_i and v_j do not share an edge in G. Further, since we consider graphs without self-loops, we can assign a special learnable vector $\mathbf{s}$ to $\mathbf{E}(v_i, v_j)$ for $i = j$. Then, we can define

$$\mathrm{atp}(\mathbf{v}) \coloneqq \big[\mathbf{E}(v_i, v_j)\big]^k_{i \geq j}\mathbf{W},$$

Table 6: Hyper-parameters for $(2,1)$-GT pre-training on PCQM4Mv2.

Parameter	Value
Learning rate	$2e-4$
Weight decay	0.1
Attention dropout	0.1
Post-attention dropout	0.1
Batch size	256
# gradient steps	2M
# warmup steps	60K
precision	bfloat16

where $\mathbf{W} \in \mathbb{R}^{d(k^2-k)/2 \times d}$ is a learnable weight matrix projecting the concatenated edge embeddings to the target dimension d. To see that the above faithfully encodes the atomic type, recall the matrix $\mathbf{K}$ over $\{1,2,3\}$, determining the atomic type that we introduced in Section 3.2. For undirected graphs, for all $i \geq j$, $\mathbf{K}_{ij} = 1$ if $\mathbf{E}(v_i, v_j) \neq \mathbf{0}$ and $\mathbf{E}(v_i, v_j) \neq \mathbf{s}$, $\mathbf{K}_{ij} = 2$ if $\mathbf{E}(v_i, v_j) = \mathbf{s}$ and $\mathbf{K}_{ij} = 3$ if $\mathbf{E}(v_i, v_j) = \mathbf{0}$. As a result, by including additional edge information into the k-GT, we simultaneously obtain atomic type embeddings atp.

B. Additional experimental details

Here, we present the hyper-parameters used during pre-training in Table 6. Further, we describe the hyper-parameter selection strategies employed for the different experiments. Across all experiments, we always select the hyper-parameters based on the best validation score and then evaluate on the test set.

For CS and PHOTO, we set the learning rate to 0.0001 and tune dropout over $\{0.5, 0.1\}$, the hidden dimension over $\{512, 1024\}$, the number of attention heads over $\{1, 2, 4\}$, the number of eigenvalues used over $\{64, 96\}$. For the node classification datasets, we disable the learning rate scheduler. We train for 100 epochs. Due to the large number of nodes and edges on these datasets, we use the SPE node-level PEs according to Equation (6) with $m = 384$.

For ALCHEMY, we only tune the learning rate. Specifically, for the $(2,1)$-GT we sweep over $\{0.001, 0.0005, 0.0003\}$ and for the $(3,1)$-GT, due to the additional computational demand over $\{0.0005, 0.0003\}$. For the $(2,1)$-GT we train for 2000 epochs and for the $(3,1)$-GT we train for 500 epochs, again due to the additional computational demand.

For the molecular classification datasets, we tune the transformers on learning rate and dropout. Specifically, we sweep the learning rate over $\{0.0005, 0.0001, 0.00005, 0.00001\}$ and dropout over $\{0.5, 0.1\}$. To ensure that the GINE baseline is representative, we invest more time for hyper-parameter tuning. Specifically, we sweep over the number of layers over $\{3, 6, 12\}$, the hidden dimension over $\{384, 768\}$. Further, we sweep the learning rate over $\{0.0005, 0.0001, 0.00005, 0.00001\}$ and dropout over $\{0.5, 0.1\}$. Finally, we sweep the choice of pooling function over $\{\text{mean}, \text{sum}\}$, used for pooling the node embeddings of the GINE into a single graph-level representation. This results in $12\times$ more hyper-parameter configurations for the GINE baseline than the transformers.

For BREC, we use a batch size of 16, 6 layers and a hidden dimension of 768 for both GNN and $(2,1)$-GT. For the $(3,1)$-GT, we use a batch size of 2, 3 layers and a hidden dimension of 192.

C. Extended notation

The *neighborhood* of a vertex v in $V(G)$ is denoted by $N(v) \coloneqq \{u \in V(G) \mid (v,u) \in E(G)\}$ and the *degree* of a vertex v is $|N(v)|$. Two graphs G and H are *isomorphic*, and we write $G \simeq H$ if there exists a bijection $\varphi \colon V(G) \to V(H)$ preserving the adjacency relation, i.e., (u,v) is in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $E(H)$. Then φ is an *isomorphism* between G and H. In the case of labeled graphs, we additionally require that $l(v) = l(\varphi(v))$ for v in $V(G)$, and similarly for attributed graphs. We further define the atomic type $\text{atp} \colon V(G)^k \to \mathbb{N}$, for $k > 0$, such that $\text{atp}(\mathbf{v}) = \text{atp}(\mathbf{w})$ for $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ if and only if the mapping $\varphi \colon V(G)^k \to V(G)^k$ where $v_i \mapsto w_i$ induces a partial isomorphism, i.e., we have $v_i = v_j \iff w_i = w_j$ and $(v_i, v_j) \in E(G) \iff (\varphi(v_i), \varphi(v_j)) \in E(G)$. Let $\mathbf{M} \in \mathbb{R}^{n \times p}$ and $\mathbf{N} \in \mathbb{R}^{n \times q}$ be two matrices then

$$\begin{bmatrix} \mathbf{M} & \mathbf{N} \end{bmatrix} \in \mathbb{R}^{n \times p+q}$$

denotes column-wise matrix concatenation. Further, let $\mathbf{M} \in \mathbb{R}^{p \times n}$ and $\mathbf{N} \in \mathbb{R}^{q \times n}$ be two matrices then

$$\begin{bmatrix} \mathbf{M} \\ \mathbf{N} \end{bmatrix} \in \mathbb{R}^{p+q \times n}$$

denotes row-wise matrix concatenation. For a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, we denote with $\mathbf{X}_i$ the i-th row vector. In the case where the rows of $\mathbf{X}$ correspond to nodes in a graph G, we use $\mathbf{X}_v$ to denote the row vector corresponding to the node $v \in V(G)$.

D. Transformers

Here, we define the (standard) *transformer* (Vaswani et al., 2017), a stack of alternating blocks of *multi-head attention* and fully-connected *feed-forward networks*. In each layer, $t > 0$, given token embeddings $\mathbf{X}^{(t-1)} \in \mathbb{R}^{L \times d}$ for L tokens, we compute

$$\mathbf{X}^{(t)} \coloneqq \mathsf{FFN}\big(\mathbf{X}^{(t-1)} + \big[h_1(\mathbf{X}^{(t-1)}) \ \dots \ h_M(\mathbf{X}^{(t-1)})\big]\mathbf{W}^O\big), \tag{7}$$

where $[\cdot]$ denotes column-wise concatenation of matrices, M is the number of heads, h_i denotes the i-th transformer head, $\mathbf{W}^O \in \mathbb{R}^{Md_v \times d}$ denotes a final projection matrix applied to the concatenated heads, and FFN denotes a feed-forward neural network applied row-wise. We define the i-th head as

$$h_i(\mathbf{X}) \coloneqq \mathsf{softmax}\big(\frac{1}{\sqrt{d_k}}\mathbf{X}\mathbf{W}^{Q,i}\big(\mathbf{X}\mathbf{W}^{K,i}\big)^T\big)\mathbf{X}\mathbf{W}^{V,i}, \tag{8}$$

where the softmax is applied row-wise and $\mathbf{W}^{Q,i}, \mathbf{W}^{K,i} \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}^{V,i} \in \mathbb{R}^{d \times d_v}$, and d_v and d are the head dimension and embedding dimension, respectively. We omit layer indices t and optional bias terms for clarity.

E. Weisfeiler–Leman

Here, we discuss additional background for the Weisfeiler–Leman hierarchy. We begin by describing the Weisfeiler–Leman algorithm, starting with the 1-WL. The 1-WL or color refinement is a well-studied heuristic for the graph isomorphism problem, originally proposed by Weisfeiler & Leman (1968).[2] Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling vertices. Formally, let $G = (V, E, \ell)$ be a labeled graph, in each iteration, $t > 0$, the 1-WL computes a vertex coloring $C^1_t \colon V(G) \to \mathbb{N}$, depending on the coloring of the neighbors. That is, in iteration $t > 0$, we set

$$C^1_t(v) \coloneqq \mathsf{RELABEL}\Big(\big(C^1_{t-1}(v), \{\!\{C^1_{t-1}(u) \mid u \in N(v)\}\!\}\big)\Big),$$

for all vertices v in $V(G)$, where $\mathsf{RELABEL}$ injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C^1_0 \coloneqq \ell$. To test if two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of vertices colored c in $\mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. It is easy to see that the algorithm cannot distinguish all non-isomorphic graphs (Cai et al., 1992). Several researchers, e.g., Babai (1979); Cai et al. (1992), devised a more powerful generalization of the former, today known as the k-dimensional Weisfeiler–Leman algorithm (k-WL), operating on k-tuples of vertices rather than single vertices.

E.1. The k-dimensional Weisfeiler–Leman algorithm

Due to the shortcomings of the 1-WL or color refinement in distinguishing non-isomorphic graphs, several researchers, e.g., Babai (1979); Cai et al. (1992), devised a more powerful generalization of the former, today known as the k-dimensional Weisfeiler-Leman algorithm, operating on k-tuples of vertices rather than single vertices.

Intuitively, to surpass the limitations of the 1-WL, the k-WL colors vertex-ordered k-tuples instead of a single vertex. More precisely, given a graph G, the k-WL colors the tuples from $V(G)^k$ for $k \geq 2$ instead of the vertices. By defining a neighborhood between these tuples, we can define a coloring similar to the 1-WL. Formally, let G be a graph, and let $k \geq 2$. In each iteration, $t \geq 0$, the algorithm, similarly to the 1-WL, computes a *coloring* $C^k_t \colon V(G)^k \to \mathbb{N}$. In the first iteration,

[2] Strictly speaking, the 1-WL and color refinement are two different algorithms. The 1-WL considers neighbors and non-neighbors to update the coloring, resulting in a slightly higher expressive power when distinguishing vertices in a given graph; see (Grohe, 2021) for details. For brevity, we consider both algorithms to be equivalent.

$t = 0$, the tuples $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ get the same color if they have the same atomic type, i.e., $C^k_0(\mathbf{v}) \coloneqq \text{atp}(\mathbf{v})$. Then, for each iteration, $t > 0$, C^k_t is defined by

$$C^k_t(\mathbf{v}) \coloneqq \mathsf{RELABEL}\big(C^k_{t-1}(\mathbf{v}), M_t(\mathbf{v})\big), \tag{9}$$

with $M_t(\mathbf{v})$ the multiset

$$M_t(\mathbf{v}) \coloneqq \big(\{\!\{C^k_{t-1}(\phi_1(\mathbf{v}, w)) \mid w \in V(G)\}\!\}, \ldots, \{\!\{C^k_{t-1}(\phi_k(\mathbf{v}, w)) \mid w \in V(G)\}\!\}\big), \tag{10}$$

and where

$$\phi_j(\mathbf{v}, w) \coloneqq (v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k).$$

That is, $\phi_j(\mathbf{v}, w)$ replaces the j-th component of the tuple $\mathbf{v}$ with the vertex w. Hence, two tuples are *adjacent* or *j-neighbors* if they are different in the j-th component (or equal, in the case of self-loops). Hence, two tuples $\mathbf{v}$ and $\mathbf{w}$ with the same color in iteration $(t-1)$ get different colors in iteration t if there exists a j in $[k]$ such that the number of j-neighbors of $\mathbf{v}$ and $\mathbf{w}$, respectively, colored with a certain color is different.

We run the k-WL algorithm until convergence, i.e., until for t in $\mathbb{N}$

$$C^k_t(\mathbf{v}) = C^k_t(\mathbf{w}) \iff C^k_{t+1}(\mathbf{v}) = C^k_{t+1}(\mathbf{w}),$$

for all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$, holds.

Similarly to the 1-WL, to test whether two graphs G and H are non-isomorphic, we run the k-WL in "parallel" on both graphs. Then, if the two graphs have a different number of vertices colored c, for c in $\mathbb{N}$, the k-WL *distinguishes* the graphs as non-isomorphic. By increasing k, the algorithm gets more powerful in distinguishing non-isomorphic graphs, i.e., for each $k \geq 2$, there are non-isomorphic graphs distinguished by $(k+1)$-WL but not by k-WL (Cai et al., 1992). In the following, we define some variants of the k-WL.

E.2. The δ-k-dimensional Weisfeiler–Leman algorithm

Malkin (2014) introduced the following variant of the k-WL, the δ-k-WL, which updates k-tuples according to

$$\begin{aligned} M^{\text{adj}}_t(\mathbf{v}) \coloneqq \big(&\{\!\{(C^{k,\text{adj}}_{t-1}(\phi_1(\mathbf{v}, w)), \text{adj}(v_1, w)) \mid w \in V(G)\}\!\}, \ldots, \\ &\{\!\{(C^{k,\text{adj}}_{t-1}(\phi_k(\mathbf{v}, w)), \text{adj}(v_k, w)) \mid w \in V(G)\}\!\}\big), \end{aligned}$$

where

$$\text{adj}(v, w) \coloneqq \begin{cases} 1, \text{ if } (v, w) \in E(G) \\ 0, \text{ otherwise,} \end{cases}$$

resulting in the coloring function $C^{k,\text{M}}_t \colon V(G)^k \to \mathbb{N}$. Morris et al. (2020) showed that this variant is slightly more powerful in distinguishing non-isomorphic graphs compared to the k-WL.

E.3. The local δ-k-dimensional Weisfeiler–Leman algorithm

Morris et al. (2020) introduced a more efficient variant of the k-WL, the *local δ-k-dimensional Weisfeiler–Leman algorithm* (δ-k-LWL), which updates k-tuples according to

$$M^{\delta}_t(\mathbf{v}) = \big(\{\!\{C^{k,\delta}_{t-1}(\phi_1(\mathbf{v}, w)) \mid w \in N(v_1)\}\!\}, \ldots, \{\!\{C^{k,\delta}_{t-1}(\phi_k(\mathbf{v}, w)) \mid w \in N(v_k)\}\!\}\big),$$

resulting in the coloring function $C^{k,\delta}_t \colon V(G)^k \to \mathbb{N}$.

E.4. The local (k, s)-dimensional Weisfeiler–Leman algorithm

Let G be a graph. Then $\#\text{comp}(G)$ denotes the number of (connected) components of G. Further, let $k \geq 1$ and $1 \leq s \leq k$, then

$$V(G)^k_s \coloneqq \{\mathbf{v} \in V(G)^k \mid \#\text{comp}(G[\mathbf{v}]) \leq s\}$$

is the set of (k, s)-*tuples* of nodes, i.e, k-tuples which induce (sub-)graphs with at most s (connected) components. In contrast to the algorithms of Equation (9), the (k, s)-LWL colors tuples from $V(G)^k_s$ instead of the entire $V(G)^k$, resulting in the coloring $C^{k,s}_t \colon V(G)^k_s \to \mathbb{N}$. For more details; see Morris et al. (2022).

E.5. Comparing k-WL variants

Let A_1 and A_2 denote k-WL-like algorithms, we write $A_1 \sqsubseteq A_2$ if A_1 distinguishes between all non-isomorphic pairs A_2 does, and $A_1 \equiv A_2$ if both directions hold. The corresponding strict relation is denoted by $\sqsubset$. We can extend these relations to neural architectures as follows. Given two non-isomorphic graphs G and H, a neural network architecture *distinguishes* them if a parameter assignment exists such that $\mathbf{h}_G \neq \mathbf{h}_H$.

E.6. The Weisfeiler–Leman hierarchy and permutation-invariant function approximation

The Weisfeiler–Leman hierarchy is a purely combinatorial algorithm for testing graph isomorphism. However, the graph isomorphism function, mapping non-isomorphic graphs to different values, is the hardest to approximate permutation-invariant function. Hence, the Weisfeiler–Leman hierarchy has strong ties to GNNs' capabilities to approximate permutation-invariant or equivariant functions over graphs. For example, Morris et al. (2019); Xu et al. (2019a) showed that the expressive power of any possible GNN architecture is limited by the 1-WL in terms of distinguishing non-isomorphic graphs. Azizian & Lelarge (2021) refined these results by showing that if an architecture is capable of simulating the k-WL (on the set of n-order graphs) and allows the application of universal neural networks on vertex features, it will be able to approximate any permutation-equivariant function below the expressive power of the k-WL; see also (Chen et al., 2019). Hence, if one shows that one architecture distinguishes more graphs than another, it follows that the corresponding GNN can approximate more functions. These results were refined in (Geerts & Reutter, 2022) for color refinement and taking into account the number of iterations of the k-WL.

E.7. A generalized adjacency matrix

Finally, we define a generalization of the adjacency matrix from nodes to k-tuples. Specifically, let G be a graph with n nodes and let $\gamma \in \{-1, 1\}$. Then, for all $j \in [k]$, the *generalized adjacency matrix* $\mathbf{A}^{(k,j,\gamma)} \in \{0,1\}^{n^k \times n^k}$ is defined as

$$\mathbf{A}^{(k,j,\gamma)}_{il} := \begin{cases} \begin{cases} 1 & \exists w \in V(G) \colon \mathbf{u}_l = \phi_j(\mathbf{u}_i, w) \wedge \mathrm{adj}(\mathbf{u}_{ij}, w) \\ 0 & \text{else} \end{cases} & \gamma = 1 \\ \begin{cases} 1 & \exists w \in V(G) \colon \mathbf{u}_l = \phi_j(\mathbf{u}_i, w) \wedge \neg\mathrm{adj}(\mathbf{u}_{ij}, w) \\ 0 & \text{else,} \end{cases} & \gamma = -1 \end{cases} \tag{11}$$

where $\mathbf{u}_i$ denotes the i-th k-tuple in a fixed but arbitrary ordering over $V(G)^k$, $\mathbf{u}_{ij}$ denotes the j-th node of $\mathbf{u}_i$ and where γ controls whether the generalized adjacency matrix considers ($\gamma = 0$) all j-neighbors, ($\gamma > 1$) j-neighbors where the swapped nodes are adjacent in G or ($\gamma < 0$) j-neighbors where the swapped nodes are not adjacent in G. With the generalized adjacency matrix as defined above we can represent the local and global j-neighborhood adjacency defined for the δ-k-WL with $\mathbf{A}^{(k,j,1)}$ and $\mathbf{A}^{(k,j,-1)}$, respectively. Further, the j-neighborhood adjacency of k-WL can be described via $\mathbf{A}^{(k,j,1)} + \mathbf{A}^{(k,j,-1)}$ and the local j-neighborhood adjacency of δ-k-LWL can be described with $\mathbf{A}^{(k,j,1)}$.

F. Structural embeddings

Before we give the missing proofs from Section 4, we develop a theoretical framework to analyze structural embeddings.

F.1. Sufficiently node and adjacency-identifying

We begin by proving the following lemma which shows an important bound for sufficiently node and adjacency-identifying matrices.

Lemma 9. *Let* $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{n \times d}$ *be matrices such that*

$$||\mathbf{P} - \mathbf{Q}||_F < \epsilon,$$

for an arbitrary but fixed $\epsilon > 0$. Let $\mathbf{P}$ be node or adjacency-identifying with projection matrices $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d\times d}$ and let

$$\tilde{\mathbf{P}} = \frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T, \text{ and}$$
$$\tilde{\mathbf{Q}} = \frac{1}{\sqrt{d_k}}(\mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T.$$

Then, there exists a monotonic strictly increasing function f such that

$$||\tilde{\mathbf{P}} - \tilde{\mathbf{Q}}||_F < f(\epsilon).$$

Proof. Our goal is to show that if the error between $\mathbf{P}$ and $\mathbf{Q}$ is bounded, so is the error between $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{Q}}$. We now show that this error can be described by a monotonic strictly increasing function f, i.e., if $\epsilon_1 < \epsilon_2$, then $f(\epsilon_1) < f(\epsilon_2)$. We will first prove the existence of f.

First note that we can write,

$$\tilde{\mathbf{P}} - \tilde{\mathbf{Q}} = \frac{1}{\sqrt{d_k}} \cdot \Big(\mathbf{P}\mathbf{W}^Q(\mathbf{P}\mathbf{W}^K - \mathbf{Q}\mathbf{W}^K)^T + (\mathbf{P}\mathbf{W}^Q - \mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T\Big).$$

Note further that we are guaranteed that

$$||\mathbf{P}||_{\text{F}} > 0 \tag{12}$$
$$||\mathbf{W}^Q||_{\text{F}} > 0 \tag{13}$$
$$||\mathbf{W}^K||_{\text{F}} > 0 \tag{14}$$

since otherwise at least one of the above matrices is zero, in which case $\tilde{\mathbf{P}} = \mathbf{0}$, which is not node or adjacency-identifying in general, a contradiction. Now we can write,

$$\begin{aligned}
||\tilde{\mathbf{P}} - \tilde{\mathbf{Q}}||_{\text{F}} &= \Big|\Big|\frac{1}{\sqrt{d_k}} \cdot \Big(\mathbf{P}\mathbf{W}^Q(\mathbf{P}\mathbf{W}^K - \mathbf{Q}\mathbf{W}^K)^T + (\mathbf{P}\mathbf{W}^Q - \mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T\Big)\Big|\Big|_{\text{F}} \\
&\overset{(a)}{\leq} \frac{1}{\sqrt{d_k}} \cdot \left(\Big|\Big|\mathbf{P}\mathbf{W}^Q(\mathbf{P}\mathbf{W}^K - \mathbf{Q}\mathbf{W}^K)^T\Big|\Big|_{\text{F}} + \Big|\Big|(\mathbf{P}\mathbf{W}^Q - \mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T\Big|\Big|_{\text{F}}\right) \\
&\overset{(b)}{\leq} \frac{1}{\sqrt{d_k}} \cdot \left(\Big|\Big|\mathbf{P}\mathbf{W}^Q\Big|\Big|_{\text{F}} \cdot \Big|\Big|(\mathbf{P}\mathbf{W}^K - \mathbf{Q}\mathbf{W}^K)^T\Big|\Big|_{\text{F}} + \Big|\Big|(\mathbf{Q}\mathbf{W}^K)^T\Big|\Big|_{\text{F}} \cdot \Big|\Big|\mathbf{P}\mathbf{W}^Q - \mathbf{Q}\mathbf{W}^Q\Big|\Big|_{\text{F}}\right) \\
&\overset{(c)}{\leq} \frac{1}{\sqrt{d_k}} \cdot \left(\Big|\Big|\mathbf{P}\mathbf{W}^Q\Big|\Big|_{\text{F}} \cdot \Big|\Big|\mathbf{P}\mathbf{W}^K - \mathbf{Q}\mathbf{W}^K\Big|\Big|_{\text{F}} + \Big|\Big|\mathbf{Q}\mathbf{W}^K\Big|\Big|_{\text{F}} \cdot \Big|\Big|\mathbf{P}\mathbf{W}^Q - \mathbf{Q}\mathbf{W}^Q\Big|\Big|_{\text{F}}\right) \\
&\overset{(d)}{\leq} \frac{1}{\sqrt{d_k}} \cdot \left(\Big|\Big|\mathbf{P}\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{W}^Q\Big|\Big|_{\text{F}} \cdot \Big|\Big|\mathbf{W}^K\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{P} - \mathbf{Q}\Big|\Big|_{\text{F}} + \Big|\Big|\mathbf{Q}\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{W}^Q\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{W}^K\Big|\Big|_{\text{F}} \cdot \Big|\Big|\mathbf{P} - \mathbf{Q}\Big|\Big|_{\text{F}}\right) \\
&= \frac{1}{\sqrt{d_k}} \cdot \Big|\Big|\mathbf{W}^Q\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{W}^K\Big|\Big|_{\text{F}} \cdot \Big|\Big|\mathbf{P} - \mathbf{Q}\Big|\Big|_{\text{F}} \cdot \Big(\Big|\Big|\mathbf{P}\Big|\Big|_{\text{F}} + \Big|\Big|\mathbf{Q}\Big|\Big|_{\text{F}}\Big) \\
&\overset{(e)}{<} \frac{\epsilon}{\sqrt{d_k}} \cdot \Big|\Big|\mathbf{W}^Q\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{W}^K\Big|\Big|_{\text{F}} \cdot \Big(\Big|\Big|\mathbf{P}\Big|\Big|_{\text{F}} + \Big|\Big|\mathbf{Q}\Big|\Big|_{\text{F}}\Big).
\end{aligned}$$

Here, we used (a) the triangle inequality, (b) the Cauchy-Schwarz inequality (c) the fact that for any matrix $\mathbf{X}$, $||\mathbf{X}||_{\text{F}} = ||\mathbf{X}^T||_{\text{F}}$, (d) again Cauchy-Schwarz and finally (e) the Lemma statement, namely that $||\mathbf{P} - \mathbf{Q}||_{\text{F}} < \epsilon$ combined with the facts in Equation (12), (13) and (14), guaranteeing that $||\tilde{\mathbf{P}} - \tilde{\mathbf{Q}}||_{\text{F}} > 0$. We set

$$f(\epsilon) \coloneqq \frac{\epsilon}{\sqrt{d_k}} \cdot \Big|\Big|\mathbf{W}^Q\Big|\Big|_{\text{F}}\Big|\Big|\mathbf{W}^K\Big|\Big|_{\text{F}} \cdot \Big(\Big|\Big|\mathbf{P}\Big|\Big|_{\text{F}} + \Big|\Big|\mathbf{Q}\Big|\Big|_{\text{F}}\Big),$$

which is clearly monotonic strictly increasing, since the norms as well as $\frac{1}{\sqrt{d_k}}$ are non-negative and Equation (12), (13) and (14) ensure that $f(\epsilon) > 0$. As a result, we can now write

$$||\tilde{\mathbf{P}} - \tilde{\mathbf{Q}}||_{\text{F}} < f(\epsilon).$$

This concludes the proof. □

We use sufficiently node or adjacency-identifying matrices for approximately recovering *weighted indicator matrices*, which we define next.

Definition 10 (Weighted indicators)**.** Let $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ be an n-dimensional binary vector. We call

$$\tilde{\mathbf{x}} \coloneqq \frac{\mathbf{x}}{\sum_{i=1}^{n} x_i},$$

the weighted indicator vector of $\mathbf{x}$. Further, let $\mathbf{X} \in \{0,1\}^{n \times n}$ be a binary matrix. Let now $\tilde{\mathbf{X}} \in \mathbb{R}^{n \times n}$ be a matrix such that the i-th row of $\tilde{\mathbf{X}}$ is the weighted indicator vector of the i-th row of $\mathbf{X}$. We call $\tilde{\mathbf{X}}$ the weighted indicator matrix of $\mathbf{X}$.

The following lemma ties (sufficiently) node or adjacency-identifying matrices and weighted indicators together.

Lemma 11. *Let $\tilde{\mathbf{P}} \in \mathbb{R}^{n \times n}$ be a matrix and let $\mathbf{X}$ be a binary matrix with weighted indicator matrix $\tilde{\mathbf{X}}$, such that for every $i, j \in [n]$,*

$$\tilde{\mathbf{P}}_{ij} = \max_k \tilde{\mathbf{P}}_{ik}$$

if, and only, if $\mathbf{X}_{ij} = 1$. Then, for a matrix $\tilde{\mathbf{Q}} \in \mathbb{R}^{n \times n}$ and all $\epsilon > 0$, there exist an $\delta > 0$ and a $b > 0$ such that if

$$\left|\left|\tilde{\mathbf{P}} - \tilde{\mathbf{Q}}\right|\right|_F < \delta,$$

then,

$$\left|\left|\mathit{softmax}\Big(b \cdot \tilde{\mathbf{Q}}\Big) - \tilde{\mathbf{X}}\right|\right|_F < \epsilon,$$

where softmax is applied row-wise.

Proof. We begin by reviewing how the softmax acts on a vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$. Let $z_{\text{max}} \coloneqq \max_i z_i$ be the maximum value in $\mathbf{z}$. Further, let $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ be a binary vector such that

$$x_i = \begin{cases} 1 & z_i = z_{\text{max}} \\ 0 & \text{else} \end{cases}.$$

Now, for a scalar $b > 0$,

$$\lim_{b \to \text{inf}} \mathsf{softmax}\Big(b \cdot \mathbf{z}\Big) = \frac{x}{\sum_{i=1}^{n} x_i},$$

i.e., as b goes to infinity, the softmax converges $b \cdot \mathbf{z}$ to the weighted indicator vector

$$\tilde{\mathbf{x}} = \frac{x}{\sum_{i=1}^{n} x_i}.$$

Let us now generalize this to matrices. Specifically, let $\tilde{\mathbf{P}} \in \mathbb{R}^{n \times n}$ be a matrix, let $\mathbf{X}$ be a binary matrix with weighted indicator matrix $\tilde{\mathbf{X}}$ such that for every $i, j \in [n]$,

$$\tilde{\mathbf{P}}_{ij} = \max_k \tilde{\mathbf{P}}_{ik}$$

if and only if $\mathbf{X}_{ij} = 1$. Then, for a scalar $b > 0$,

$$\lim_{b \to \text{inf}} \mathsf{softmax}\Big(b \cdot \tilde{\mathbf{P}}\Big) = \tilde{\mathbf{X}},$$

which follows from the fact that the softmax is applied independently to each row and each row of $\tilde{\mathbf{X}}$ is a weighted indicator vector. We now show the proof statement. First, note that for any $b > 0$ we can choose a $\delta < f(b)$, where $f : \mathbb{R} \to \mathbb{R}$ is some strictly monotonically *decreasing* function of b that shrinks faster than linear, e.g., $f(b) = \frac{1}{b^2}$. This function is well-defined since by assumption $b > 0$. As a result an increase in b implies a non-linearly growing decrease of

$$\left|\left|\mathsf{softmax}\Big(b \cdot \tilde{\mathbf{P}}\Big) - \mathsf{softmax}\Big(b \cdot \tilde{\mathbf{Q}}\Big)\right|\right|_{\text{F}}$$

and thus,

$$\lim_{b \to \text{inf}} \mathsf{softmax}\Big(b \cdot \tilde{\mathbf{Q}}\Big) = \tilde{\mathbf{X}},$$

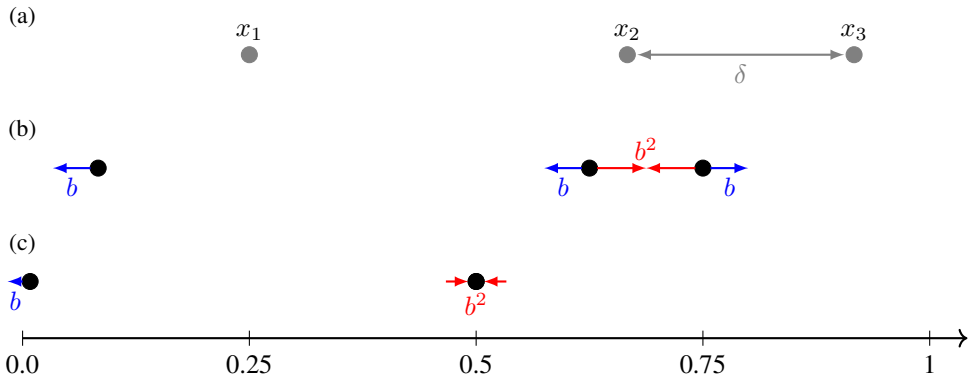

Figure 2: Visual explanation of the "opposing forces" in Lemma 11. In (a) **before softmax**: We consider three numbers x_1, x_2 and x_3, where x_2 and x_3 are less than δ apart. In (b) **after softmax**: An increase in b (blue) pushes the maximum value x_3 away from x_1 and x_2. However, the approximation with δ acts stronger (red). As a result, x_1 gets pushed closer to 0, but x_2 and x_3 get pushed closer. In (c) **after softmax**: Further increasing b makes x_1 converge to 0, but the approximation with δ pushes x_2 and x_3 closer together, and the softmax maps both values approximately to $\frac{1}{2}$ (here depicted with the same dot). Hence, with a sufficiently close approximation, we can approximate the weighted indicator matrix $\tilde{\mathbf{X}}$.

and we have that for all $\epsilon > 0$ there exists a $b > 0$ and a $\delta < f(b)$ such that

$$\left|\left|\mathsf{softmax}\left(b \cdot \tilde{\mathbf{Q}}\right) - \tilde{\mathbf{X}}\right|\right|_{\mathrm{F}} < \epsilon.$$

This concludes the proof. □

In the above proof, the approximation with δ and the scaling with b act as two "opposing forces". The proof then chooses b such that ϵ acts stronger than b and hence with $b \to \inf$, the approximation converges to the weighted indicator matrix; see Figure 2 for a visual explanation of this concept.

From the definitions of sufficiently node and adjacency-identifying matrices, we can prove the following statement.

Lemma 12. *Let G be a graph with adjacency matrix $\mathbf{A}(G)$ and degree matrix $\mathbf{D}$. Let $\mathbf{Q}$ be a sufficiently adjacency-identifying matrix. Then, there exists a $b > 0$ and projection matrices $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d \times d}$ with*

$$\tilde{\mathbf{Q}} \coloneqq \frac{1}{\sqrt{d_k}}(\mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T$$

such that

$$\left|\left|\mathit{softmax}\left(b \cdot \tilde{\mathbf{Q}}\right) - \mathbf{D}^{-1}\mathbf{A}(G)\right|\right|_F < \epsilon,$$

for all $\epsilon > 0$.

Proof. Note that $\mathbf{D}^{-1}\mathbf{A}(G)$ is a weighted indicator matrix, since $\mathbf{A}(G)$ has binary entries and left-multiplication with $\mathbf{D}^{-1}$ results in dividing every element in row i of $\mathbf{A}(G)$ by the number of 1's in row i of $\mathbf{A}(G)$, or formally,

$$\left(\mathbf{D}^{-1}\mathbf{A}(G)\right)_i = \left[\frac{\mathbf{A}(G)_{i1}}{\sum_{j=1}^n \mathbf{A}(G)_{ij}} \quad \cdots \quad \frac{\mathbf{A}(G)_{in}}{\sum_{j=1}^n \mathbf{A}(G)_{ij}}\right],$$

where we note that

$$\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}(G)_{ij}.$$

Further, since $\mathbf{Q}$ is sufficiently adjacency-identifying, there exists an adjacency-identifying matrix $\mathbf{P}$ and projection matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$, such that for

$$\tilde{\mathbf{P}} = \frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T,$$

it holds that

$$\tilde{\mathbf{P}}_{ij} = \max_k \tilde{\mathbf{P}}_{ik},$$

if and only if $\mathbf{A}(G) = 1$. Recall the definition of sufficiently adjacency-identifying, namely that we can choose projection matrices such that

$$||\mathbf{P} - \mathbf{Q}||_{\mathrm{F}} < \epsilon_0,$$

for all ϵ_0. Then, according to Lemma 9, for matrix

$$\tilde{\mathbf{Q}} = \frac{1}{\sqrt{d_k}}(\mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T,$$

it holds that

$$\left|\left|\tilde{\mathbf{P}} - \tilde{\mathbf{Q}}\right|\right|_{\mathrm{F}} < f(\epsilon_0),$$

where f is a strictly monotonically increasing function of ϵ_0. We can then apply Lemma 11 to $\tilde{\mathbf{P}}$, $\tilde{\mathbf{Q}}$, $\mathbf{A}(G)$ as the binary matrix and $\mathbf{D}^{-1}\mathbf{A}(G)$ as its weighted indicator matrix and, for every ϵ, choose ϵ_0 small enough such that there exists a $b > 0$ with

$$\left|\left|\mathsf{softmax}\Big(b \cdot \tilde{\mathbf{Q}}\Big) - \mathbf{D}^{-1}\mathbf{A}(G)\right|\right|_{\mathrm{F}} < \epsilon.$$

This concludes the proof. □

Finally, we generalize the above result to arbitrary k via the generalized adjacency matrix.

Lemma 13 (Approximating the generalized adjacency matrix)**.** *Let G be graph with n nodes and let $k > 1$. Let further, $\mathbf{Q} \in \mathbb{R}^{n \times d/k}$ be sufficiently node and adjacency-identifying structural embeddings. Let $\mathbf{v} \coloneqq (v_1, \ldots, v_k) \in V(G)^k$ be the i-th and let $\mathbf{u} \coloneqq (u_1, \ldots, u_k) \in V(G)^k$ be the l-th k-tuple in a fixed but arbitrary ordering over $V(G)^k$. Let $\tilde{\mathbf{A}}^{(k,j,\gamma)}$ denote the weighted indicator matrix of the generalized adjacency matrix in Equation (11). Let $\mathbf{Z} \in \mathbb{R}^{n^k \times n^k}$ be the unnormalized attention matrix such that*

$$\mathbf{Z}_{il} \coloneqq \frac{1}{\sqrt{d_k}}\big[\mathbf{Q}(v_j)\big]_{j=1}^{k}\mathbf{W}^Q\big(\big[\mathbf{Q}(u_j)\big]_{j=1}^{k}\mathbf{W}^K\big)^T,$$

with projection matrices $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d \times d}$. Then, for every $j \in [k]$ and every $\gamma \in \{-1, 1\}$, there exist projection matrices $\mathbf{W}^Q, \mathbf{W}^K$ such that for all $i \in [n^k]$

$$\left\|\mathsf{softmax}\Big(\begin{bmatrix}\mathbf{Z}_{i1} & \ldots & \mathbf{Z}_{in^k}\end{bmatrix}\Big) - \tilde{\mathbf{A}}_i^{(k,j,\gamma)}\right\|_F < \varepsilon,$$

for any $\varepsilon > 0$.

Proof. Our proof strategy is to first construct the unnormalized attention matrix from a node- and adjacency-identifying matrix and then to invoke Lemma 11 to relax the approximation to the *sufficiently* node- and adjacency-identifying $\mathbf{Q}$.

First, by definition of sufficiently node and adjacency-identifying matrices, the existence of $\mathbf{Q}$ implies the existence of a node and adjacency-identifying matrix $\mathbf{P}$. We will now give projection matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ such that

$$\left\|\mathsf{softmax}\Big(\begin{bmatrix}\mathbf{Z}^*_{i1} & \ldots & \mathbf{Z}^*_{in^k}\end{bmatrix}\Big) - \tilde{\mathbf{A}}_i^{(k,j,\gamma)}\right\|_F < \varepsilon. \tag{15}$$

for any $\varepsilon > 0$, with

$$\mathbf{Z}^*_{il} \coloneqq \frac{1}{\sqrt{d_k}}\big[\mathbf{P}(v_j)\big]_{j=1}^{k}\mathbf{W}^Q\big(\big[\mathbf{P}(u_j)\big]_{j=1}^{k}\mathbf{W}^K\big)^T,$$

where we recall that $\mathbf{v} = (v_1, \ldots, v_k) \in V(G)^k$ is the i-th and $\mathbf{u} = (u_1, \ldots, u_k) \in V(G)^k$ is the l-th k-tuple in a fixed but arbitrary ordering over $V(G)^k$. To show Equation (15), we expand sub-matrices in $\mathbf{W}^Q$ and $\mathbf{W}^K$ for each j, i.e., we write

$$\mathbf{W}^Q = \begin{bmatrix}\mathbf{W}^{Q,1} \\ \vdots \\ \mathbf{W}^{Q,k}\end{bmatrix}$$

and

$$\mathbf{W}^K = \begin{bmatrix} \mathbf{W}^{K,1} \\ \vdots \\ \mathbf{W}^{K,k} \end{bmatrix},$$

where for all $j \in [k]$, $\mathbf{Q}(v_j)$ is projected by $\mathbf{W}^{Q,j}$ and $\mathbf{Q}(u_j)$ is projected by $\mathbf{W}^{K,j}$. We further expand the sub-matrices of each $\mathbf{W}^{Q,j}$ and $\mathbf{W}^{K,j}$, writing

$$\mathbf{W}^{Q,j} = \begin{bmatrix} \mathbf{W}_N^{Q,j} \\ \mathbf{W}_A^{Q,j} \end{bmatrix}$$

and

$$\mathbf{W}^{K,j} = \begin{bmatrix} \mathbf{W}_N^{K,j} \\ \mathbf{W}_A^{K,j} \end{bmatrix}.$$

Since $\mathbf{P}$ is node-identifying, there exists projection matrices $\mathbf{W}_N^{Q,*}$ and $\mathbf{W}_N^{K,*}$ such that

$$p_{ilj} \coloneqq \frac{1}{\sqrt{d_k}} \mathbf{P}^* \mathbf{W}_N^{Q,*} (\mathbf{P}^* \mathbf{W}_N^{K,*})^T$$

is maximal if and only if $\mathbf{u}_{i,j} = \mathbf{u}_{l,j}$ is the same node. Further, if $\mathbf{P}$ is adjacency-identifying, there exists projection matrices $\mathbf{W}_A^{Q,*}$ and $\mathbf{W}_A^{K,*}$ such that

$$q_{ilj} \coloneqq \frac{1}{\sqrt{d_k}} \mathbf{P}^* \mathbf{W}_A^{Q,*} (\mathbf{P}^* \mathbf{W}_A^{K,*})^T,$$

is maximal if and only if nodes $\mathbf{u}_{i,j}$ and $\mathbf{u}_{l,j}$ share an edge in G. Consequently, we also know that then $-q_{ilj}$ is maximal if and only if nodes $\mathbf{u}_{i,j}$ and $\mathbf{u}_{l,j}$ do not share an edge in G. Note that because we assume that G has no self-loops, $q_{iij} = 0$ for all $i \in [n]$ and all $j \in [k]$. Now, let us write out the unnormalized attention score $\mathbf{Z}_{il}^*$ as

$$\begin{aligned}
\mathbf{Z}_{il}^* &= \frac{1}{\sqrt{d_k}} \big[\mathbf{P}(v_j)\big]_{j=1}^k \mathbf{W}^Q \big(\big[\mathbf{P}(u_j)\big]_{j=1}^k \mathbf{W}^K\big)^T \\
&= \frac{1}{\sqrt{d_k}} \sum_{j=1}^k \mathbf{P}(v_j) \mathbf{W}^{Q,j} \big(\mathbf{P}(u_j) \mathbf{W}^{K,j}\big)^T \\
&= \frac{1}{\sqrt{d_k}} \sum_{j=1}^k \mathbf{P}(v_j) \mathbf{W}_N^{Q,j} \big(\mathbf{P}(u_j) \mathbf{W}_N^{K,j}\big)^T + \frac{1}{\sqrt{d_k}} \sum_{j=1}^k \mathbf{P}(v_j) \mathbf{W}_A^{Q,j} \big(\mathbf{P}(u_j) \mathbf{W}_A^{K,j}\big)^T,
\end{aligned}$$

where in the second line we simply write out the dot-product and in the last line we split the sum to distinguish node- and adjacency-identifying terms.

Now for all γ and a fixed j, we set $\mathbf{W}_N^{Q,j} = \mathbf{0}$ and $\mathbf{W}_N^{K,j} = \mathbf{0}$ and $\mathbf{W}_N^{Q,o} = b \cdot \mathbf{W}_N^{Q,*}$ and $\mathbf{W}_N^{K,o} = \mathbf{W}_N^{K,*}$ for $o \neq j$, for some $b > 0$ that we need to be arbitary to use it later in Lemma 11.

We now distinguish between the different choices of γ. In the first case, let $\gamma > 0$. Then, we set $\mathbf{W}_A^{Q,j} = \mathbf{W}_A^{Q,*}$ and $\mathbf{W}_A^{K,j} = \mathbf{W}_A^{K,*}$. In the third case, $\gamma < 0$. Then, we set $\mathbf{W}_A^{Q,j} = -\mathbf{W}_A^{Q,*}$ and $\mathbf{W}_A^{K,j} = \mathbf{W}_A^{K,*}$.

Then, in all cases, for tuples $\mathbf{u}_i$ and $\mathbf{u}_l$,

$$\mathbf{Z}_{il}^* = \gamma \cdot q_{ilj} + \sum_{o \neq j} b \cdot p_{ilo}. \tag{16}$$

Note that b is the same for all pairs $\mathbf{u}_i, \mathbf{u}_l$ and the dot-product is positive. We again distinguish between two cases. In the first case, let $\gamma > 0$. Then, the above sum attains its maximum value if and only if

$$\forall o \neq j \colon \mathbf{u}_{i,o} = \mathbf{u}_{l,o} \wedge A_{ilj} = 1.$$

Now, since $\mathbf{u}_{i,o}$ and $\mathbf{u}_{l,o}$ denote nodes at the o-th component of $\mathbf{u}_i$ and $\mathbf{u}_l$, respectively, the above statement is equivalent to saying that $\mathbf{u}_l$ is a j-neighbor of $\mathbf{u}_i$ and that $\mathbf{u}_l$ is adjacent to $\mathbf{u}_i$. In the second case, let $\gamma < 0$. Then, the above sum attains its maximum value if and only if

$$\forall o \neq j \colon \mathbf{u}_{i,o} = \mathbf{u}_{l,o} \wedge A_{ilj} = 0.$$

Again, since $\mathbf{u}_{i,o}$ and $\mathbf{u}_{l,o}$ denote nodes at the o-th component of $\mathbf{u}_i$ and $\mathbf{u}_l$, respectively, the above statement is equivalent to saying that $\mathbf{u}_l$ is a j-neighbor of $\mathbf{u}_i$ and that $\mathbf{u}_l$ is *not* adjacent to $\mathbf{u}_i$.

Now, recall the generalized adjacency matrix in Equation (11) as

$$\mathbf{A}_{il}^{(k,j,\gamma)} \coloneqq \begin{cases} \begin{cases} 1 & \exists w \in V(G) \colon \mathbf{u}_l = \phi_j(\mathbf{u}_i, w) \wedge \mathrm{adj}(\mathbf{u}_{ij}, w) \\ 0 & \text{else} \end{cases} & \gamma = 1 \\ \begin{cases} 1 & \exists w \in V(G) \colon \mathbf{u}_l = \phi_j(\mathbf{u}_i, w) \wedge \neg\mathrm{adj}(\mathbf{u}_{ij}, w) \\ 0 & \text{else.} \end{cases} & \gamma = -1 \end{cases}$$

Then, we can say that for our construction of $\mathbf{Z}^*$, for all $i, l \in [n^k]$

$$\mathbf{Z}^*_{il} = \max_k \mathbf{Z}^*_{ik}$$

if and only if $\mathbf{A}_{il}^{k,j,\gamma} = 1$. Consequently, we can apply Lemma 11 to $\mathbf{Z}^*$, $\mathbf{Z}$, $\mathbf{A}^{k,j,\gamma}$ as the binary matrix and $\tilde{\mathbf{A}}^{k,j,\gamma}$ as its weighted indicator matrix and obtain that for each $\epsilon > 0$ there exists a $b > 0$ such that

$$\left\| \mathsf{softmax}\Big(\begin{bmatrix} \mathbf{Z}_{i1} & \dots & \mathbf{Z}_{in^k} \end{bmatrix} \Big) - \tilde{\mathbf{A}}_i^{k,j,\gamma} \right\|_F < \varepsilon.$$

This completes the proof. □

F.2. adjacency-identifying via graph Laplacian

Here, we show a few results for how factorizations of the (normalized) graph Laplacian can be adjacency-identifying.

Lemma 14. *Let G be a graph with graph Laplacian $\mathbf{L}$. Then, a matrix $\mathbf{P}$ is node- and adjacency-identifying if there exists matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ such that*

$$\frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T = \mathbf{L}.$$

Proof. Note that the graph Laplacian is node-identifying, since all off-diagonal elements are ≤ 0 and the diagonal is always > 0 since we consider graphs G without self-loops and without isolated nodes. Note further, that if there exists matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ such that

$$\frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T = \mathbf{L},$$

then there also exist matrix $\mathbf{W}^Q_* = -\mathbf{W}^Q$ such that

$$\frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q_*)(\mathbf{P}\mathbf{W}^K)^T = -\mathbf{L}.$$

Now, note that the negative graph Laplacian is $-\mathbf{L} = \mathbf{A}(G) - \mathbf{D}$. Because we subtract the degree matrix from the adjacency matrix, the maximum element of each row of the negative graph Laplacian is 1. Since $\mathbf{D}$ is diagonal, for each row i and each column j,

$$-\mathbf{L}_{ij} = 1 \iff \mathbf{A}(G)_{ij} = 1,$$

for $i \neq j$. Further, in the case where $i = j$,

$$-\mathbf{L}_{ij} \leq 0,$$

since we consider graphs G without self-loops. Hence, we obtain that

$$-\mathbf{L}_{ij} = \max_k(-\mathbf{L}_{ik})$$

if and only if $\mathbf{A}(G)_{ij} = 1$. This shows the statement. □

Lemma 15. *Let G be a graph with graph Laplacian $\mathbf{L}$. Then, a matrix $\mathbf{P}$ is node and adjacency-identifying if*

$$\mathbf{L} = \mathbf{P}\mathbf{P}^T.$$

Proof. Our goal is to show that there exist matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$, such that

$$\frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T = \mathbf{L},$$

and then to invoke Lemma 14. We set

$$\mathbf{W}^Q = \sqrt{d_k}\mathbf{I}$$
$$\mathbf{W}^K = \mathbf{I},$$

where $\mathbf{I}$ is the identity matrix. Then,

$$\frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T = \mathbf{P}\mathbf{P}^T = \mathbf{L}.$$

This shows the statement. □

Lemma 16. *Let G be a graph with graph Laplacian $\mathbf{L}$. Then, a matrix $\begin{bmatrix}\mathbf{P} & \mathbf{Q}\end{bmatrix}$ is node- and adjacency-identifying if*

$$\mathbf{L} = \mathbf{P}\mathbf{Q}^T.$$

Proof. Our goal is to show that there exist matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$, such that

$$\frac{1}{\sqrt{d_k}}(\mathbf{P}\mathbf{W}^Q)(\mathbf{P}\mathbf{W}^K)^T = \mathbf{L},$$

and then to invoke Lemma 14. We set

$$\mathbf{W}^Q = \begin{bmatrix}\sqrt{d_k}\mathbf{I} & \mathbf{0}\end{bmatrix}$$
$$\mathbf{W}^K = \begin{bmatrix}\mathbf{0} & \sqrt{d_k}\mathbf{I}\end{bmatrix},$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is the all-zero matrix. Then,

$$\frac{1}{\sqrt{d_k}}\left(\begin{bmatrix}\mathbf{P} & \mathbf{Q}\end{bmatrix}\mathbf{W}^Q\right)\left(\begin{bmatrix}\mathbf{P} & \mathbf{Q}\end{bmatrix}\mathbf{W}^K\right)^T = \mathbf{P}\mathbf{Q}^T = \mathbf{L}.$$

This shows the statement. □

For the next two lemmas, we first briefly define permutation matrices as binary matrices $\mathbf{M}$ that are right-stochastic, i.e., its rows sum to 1, and such that each column of $\mathbf{M}$ is 1 at exactly one position and 0 elsewhere. It is well know that for permutation matrices $\mathbf{M}$ it holds that $\mathbf{M}^T\mathbf{M} = \mathbf{M}\mathbf{M}^T = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. We now state the following lemmas.

Lemma 17. *Let G be a graph with graph Laplacian $\mathbf{L}$. Let $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ be the eigendecomposition of $\mathbf{L}$. Then, for any permutation matrix $\mathbf{M}$, the matrix $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ is adjacency-identifying.*

Proof. We have that

$$\begin{aligned}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}(\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M})^T &= \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}\mathbf{M}^T\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{U}^T\\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T\\ &= \mathbf{L}.\end{aligned}$$

Hence, by Lemma 15, $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ is adjacency-identifying. □

Lemma 18. *Let G be a graph with graph Laplacian $\mathbf{L}$ and normalized graph Laplacian $\tilde{\mathbf{L}}$. Let $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ be the eigendecomposition of $\mathbf{L}$ and let $\mathbf{D}$ denote the degree matrix. Then, for any permutation matrix $\mathbf{M}$ the matrix $\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ is adjacency-identifying.*

Proof. We have that

$$\begin{aligned}\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}(\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M})^T &= \mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}\mathbf{M}^T\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{U}^T\mathbf{D}^{\frac{1}{2}}\\ &= \mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T\mathbf{D}^{\frac{1}{2}}\\ &= \mathbf{D}^{\frac{1}{2}}\tilde{\mathbf{L}}\mathbf{D}^{\frac{1}{2}}\\ &= \mathbf{D}^{\frac{1}{2}}(\mathbf{I}-\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{D}^{\frac{1}{2}}\\ &= \mathbf{D}-\mathbf{A}=\mathbf{L}.\end{aligned}$$

Hence, by Lemma 15, $\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$ is node or adjacency-identifying. □

F.3. LPE

Here, we show that the node-level PEs LPE as defined in Equation (5) are sufficiently node- and adjacency-identifying. We begin with the following useful lemma.

Lemma 19. *Let $\mathbf{P}\in\mathbb{R}^{n\times d}$ be structural embeddings with sub-matrices $\mathbf{Q}_1\in\mathbb{R}^{n\times d'}$ and $\mathbf{Q}_2\in\mathbb{R}^{n\times d''}$, i.e.,*

$$\mathbf{P}=\begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix},$$

*where $d=d'+d''$. If one of $\mathbf{Q}_1,\mathbf{Q}_2$ is (sufficiently) node-identifying, then $\mathbf{P}$ is (sufficiently) node-identifying. If one of $\mathbf{Q}_1,\mathbf{Q}_2$ is (sufficiently) adjacency-identifying, then $\mathbf{P}$ is (sufficiently) adjacency-identifying. If $\mathbf{Q}_1$ is (sufficiently) node-identifying and $\mathbf{Q}_2$ is (sufficiently) adjacency-identifying or vice versa, then $\mathbf{P}$ is (sufficiently) node **and** adjacency-identifying.*

Proof. Let a sub-matrix $\mathbf{Q}$ be (sufficiently) node- or adjacency-identifying. Let $\mathbf{W}^{Q,Q}$ and $\mathbf{W}^{K,Q}$ be the corresponding projection matrices with which $\mathbf{Q}$ is (sufficiently) node- or adjacency-identifying. Then, if $\mathbf{Q}=\mathbf{Q}_1$, we define

$$\mathbf{W}^Q=\begin{bmatrix}\mathbf{W}^{Q,Q} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$$
$$\mathbf{W}^K=\begin{bmatrix}\mathbf{W}^{Q,K} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$$

and if $\mathbf{Q}=\mathbf{Q}_2$, we define

$$\mathbf{W}^Q=\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{W}^{Q,Q} & \mathbf{0}\end{bmatrix}$$
$$\mathbf{W}^K=\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{W}^{Q,K} & \mathbf{0}\end{bmatrix}.$$

In both cases, we have that

$$\mathbf{P}\mathbf{W}^Q=\begin{bmatrix}\mathbf{Q}\mathbf{W}^{Q,Q} & \mathbf{0}\end{bmatrix}$$
$$\mathbf{P}\mathbf{W}^K=\begin{bmatrix}\mathbf{Q}\mathbf{W}^{Q,K} & \mathbf{0}\end{bmatrix}$$

and consequently,

$$\mathbf{P}\mathbf{W}^Q(\mathbf{P}\mathbf{W}^K)^T=\mathbf{Q}\mathbf{W}^{Q,Q}(\mathbf{Q}\mathbf{W}^{Q,K})^T.$$

Hence if $\mathbf{Q}_1$ is (sufficiently) node-identifying, then $\mathbf{P}$ is (sufficiently) node-identifying. If $\mathbf{Q}_2$ is (sufficiently) node-identifying, then $\mathbf{P}$ is (sufficiently) node-identifying. If $\mathbf{Q}_1$ is (sufficiently) adjacency-identifying, then $\mathbf{P}$ is (sufficiently) adjacency-identifying. If $\mathbf{Q}_2$ is (sufficiently) adjacency-identifying, then $\mathbf{P}$ is (sufficiently) adjacency-identifying. If $\mathbf{Q}_1$ is (sufficiently) node-identifying and $\mathbf{Q}_2$ is (sufficiently) adjacency-identifying or vice versa, then $\mathbf{P}$ is (sufficiently) node **and** adjacency-identifying. This concludes the proof. □

We will now prove a slightly more general statement than Theorem 7.

Theorem 20 (Slightly more general than Theorem 7 in main text)**.** *Structural embeddings with LPE according to Equation* (5) *as node-level PEs with embedding dimension d are sufficiently node- and adjacency-identifying, irrespective of whether the underlying Laplacian is normalized or not. Further, for graphs with n nodes, the statement holds for $d \geq (2n+1)$.*

Proof. We begin by stating that the domain of LPE is compact since eigenvectors are unit-norm and eigenvalues are bounded by twice the maximum node degree for graphs without self-loops (Anderson & Morley, 1985). Further, the domain of the structural embeddings is compact, since LPE is compact and the node degrees are finite. Hence, the domain of deg is compact.

Let G be a graph with graph Laplacian $\mathbf{L}$. Let $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ be the eigendecomposition of $\mathbf{L}$, where the i-th column of $\mathbf{U}$ contains the i-th eigenvector, denoted $\mathbf{v}_i$, and the i-th diagonal entry of $\mathbf{\Sigma}$ contains the i-th eigenvalue, denoted λ_i.

Recall that structural embeddings with LPE as node-level PEs are defined as

$$\mathbf{P}(v) = \mathsf{FFN}(\deg(v) + \mathrm{LPE}(v)),$$

for all $v \in V(G)$. Since the domains of both deg and LPE are compact, there exist parameters for both of these embeddings such that without loss of generality we may assume that for $v \in V(G)$,

$$\deg(v) + \mathrm{LPE}(v) = \begin{bmatrix}\deg(v) & \mathrm{LPE}(v)\end{bmatrix} \in \mathbb{R}^d$$

and that deg and LPE are instead embedded into some smaller p-dimensional and s-dimensional sub-spaces, respectively, where it holds that $d = p + s$.

To show node **and** adjacency identifiability, we divide up the s-dimensional embedding space of LPE into two $s/2$-dimensional sub-spaces node and adj, i.e., we write

$$\mathrm{LPE}(v) = \begin{bmatrix}\mathrm{node}(v) & \mathrm{adj}(v)\end{bmatrix} \in \mathbb{R}^{n \times s},$$

where

$$\mathrm{node}(v) = \rho_{\mathrm{node}}\Big(\sum_{j=1}^{k} \phi_{\mathrm{node}}(\mathbf{v}_{ji}, \lambda_j + \epsilon_j)\Big)$$
$$\mathrm{adj}(v) = \rho_{\mathrm{adj}}\Big(\sum_{j=1}^{k} \phi_{\mathrm{adj}}(\mathbf{v}_{ji}, \lambda_j + \epsilon_j)\Big),$$

and where we have chosen ρ to be a sum over the first dimension of its input, followed by FFN ρ_{node} and ρ_{adj}, for node and adjacency, respectively. Note that we have written out Equation (5) for a single node v. For convenience, we also fix an arbitrary ordering over the nodes in $V(G)$ and define

$$\mathbf{Q}^D = \begin{bmatrix}\deg(v_1)\\ \vdots \\ \deg(v_n)\end{bmatrix}$$
$$\mathbf{Q}^N = \begin{bmatrix}\mathrm{node}(v_1)\\ \vdots \\ \mathrm{node}(v_n)\end{bmatrix}$$
$$\mathbf{Q}^A = \begin{bmatrix}\mathrm{adj}(v_1)\\ \vdots \\ \mathrm{adj}(v_n)\end{bmatrix},$$

where v_i is the i-th node in our ordering. Further, note that node and adj are DeepSets (Zaheer et al., 2017) over the set

$$M_i \coloneqq \{(\mathbf{v}_{ji}, \lambda_j + \epsilon_j)\}_{j=1}^{k},$$

where $\mathbf{v}_j$ is again the j-th eigenvector with corresponding eigenvalue λ_j and ϵ_j is a learnable scalar. With DeepSets we can universally approximate permutation invariant functions. We will use this fact in the following, where we show that there exists

a parameterization of $\mathbf{Q}^N$ and $\mathbf{Q}^A$ such that $\mathbf{Q}^N$ is sufficiently node-identifying and $\mathbf{Q}^A$ is sufficiently adjacency-identifying. Then, it follows from Lemma 19 that $\mathbf{P}$ is sufficiently node and adjacency-identifying. We will further use $\mathbf{Q}^D$ later in the proof.

We begin by showing that $\mathbf{Q}^N$ is sufficiently node-identifying. Observe that the eigenvector matrix $\mathbf{U}$ is already node-identifying since $\mathbf{U}$ forms an orthonormal basis and hence $\mathbf{U}\mathbf{U}^T$ is the n-dimensional identity matrix $\mathbf{I}$. Clearly for $\mathbf{I}$ it holds that

$$\mathbf{I}_{ij} = \max_k \mathbf{I}_{ik},$$

if and only if $i = j$. Moreover, let $\mathbf{M}$ be any permutation matrix, i.e., each column of $\mathbf{M}$ is 1 at exactly one position and 0 else and the rows of $\mathbf{M}$ sum to 1. Then,

$$\mathbf{U}\mathbf{M}\mathbf{M}^T\mathbf{U}^T = \mathbf{U}\mathbf{U}^T = \mathbf{I}$$

is also node-identifying. We will now approximate $\mathbf{U}\mathbf{M}$ for some $\mathbf{M}$ with node arbitrarily close. Specifically, for the i-th node v_i in our node ordering, we choose ρ_{node} and ϕ_{node} such that

$$\left|\left|\text{node}(v_i) - \mathbf{U}_i\mathbf{M}\right|\right|_{\text{F}} < \epsilon,$$

for all $\epsilon > 0$ and all i. Since DeepSets can universally approximate permutation invariant functions, it remains to show that there exists a permutation invariant function f such that $f(M_i) = \mathbf{U}_i\mathbf{M}$ for all i and for some $\mathbf{M}$. To this end, note that for a graph G, there are only at most n unique eigenvalues of the corresponding (normalized) graph Laplacian. Hence, we can choose ϵ_j such that $\lambda_j + \epsilon_j$ is unique for each unique j. In particular, let

$$\epsilon_j = j \cdot \delta,$$

where we choose $\delta < \min_{l,o} |\lambda_l - \lambda_o| > 0$, i.e., the smallest non-zero difference between two eigenvalues. We now define f as

$$f(\{(\mathbf{v}_{ji}, \lambda_j + \epsilon_j)\}_{j=1}^k) = \begin{bmatrix}\mathbf{v}_{1i} & \dots & \mathbf{v}_{ki}\end{bmatrix} = \mathbf{U}_i\mathbf{M},$$

where the order of the components is according to the sorted $\lambda_j + \epsilon_j$ in ascending order. This order is reflected in some permutation matrix $\mathbf{M}$. Hence, f is permutation invariant and can be approximated by a DeepSet arbitrarily close. As a result, we have

$$\left|\left|\text{node}(v_i) - \mathbf{U}_i\mathbf{M}\right|\right|_{\text{F}} < \epsilon,$$

for all i and an arbitrarily small $\epsilon > 0$. In matrix form, we have that

$$\left|\left|\mathbf{U}\mathbf{M} - \mathbf{Q}^N\right|\right|_{\text{F}} < \epsilon$$

and since $\mathbf{U}\mathbf{M}$ is node-identifying, we can invoke Lemma 9, to conclude that $\mathbf{Q}^N$ is sufficiently node-identifying. As a result, $\mathbf{P}$ has a sufficiently node-identifying sub-space and is thus, also sufficiently node-identifying according to Lemma 19.

We continue by showing that $\mathbf{Q}^A$ is sufficiently adjacency-identifying. Note that according to Lemma 17, $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$ is adjacency-identifying. We will now approximate $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$ with adj arbitrarily close. Specifically, for the i-th node v_i in our node ordering, we choose ρ_{adj} and ϕ_{adj} such that

$$\left|\left|\text{adj}(v_i) - \mathbf{U}\mathbf{\Sigma}_i^{\frac{1}{2}}\right|\right|_{\text{F}} < \epsilon,$$

for all $\epsilon > 0$ and all i. To this end, we first note that right-multiplication of $\mathbf{U}$ by $\mathbf{\Sigma}^{\frac{1}{2}}$ is equal to multiplying the i-th column of $\mathbf{U}$, i.e., the eigenvector $\mathbf{v}_i$, with the j-th diagonal element of $\mathbf{\Sigma}^{\frac{1}{2}}$, i.e., $\sqrt{\lambda_j}$. Hence, for the j-node v_j it holds

$$\mathbf{U}\mathbf{\Sigma}_i^{\frac{1}{2}} = \begin{bmatrix}\mathbf{v}_{1j} \cdot \sqrt{\lambda_1} & \dots & \mathbf{v}_{nj} \cdot \sqrt{\lambda_n}\end{bmatrix} \in \mathbb{R}^n,$$

where $\mathbf{v}_{ij}$ denotes the j-th component of $\mathbf{v}_i$. Since DeepSets can universally approximate permutation invariant functions, it remains to show that there exists a permutation invariant function f such that $f(M_i) = \mathbf{U}\mathbf{\Sigma}_i^{\frac{1}{2}}\mathbf{M}$ for all i and for some $\mathbf{M}$. To this end, note that for a graph G, there are only at most n unique eigenvalues of the corresponding (normalized) graph Laplacian. Hence, we can choose ϵ_j such that $\lambda_j + \epsilon_j$ is unique for each unique j. In particular, let

$$\epsilon_j = j \cdot \delta,$$

where we choose $\delta < \min_{l,o} |\lambda_l - \lambda_o| > 0$, i.e., the smallest non-zero difference between two eigenvalues. We now define f as

$$f(\{(\mathbf{v}_{ji}, \lambda_j + j \cdot \delta)\}_{j=1}^{k}) = \begin{bmatrix} \mathbf{v}_{1i} \cdot \sqrt{\lambda_1 + \epsilon_1 + \delta} & \dots & \mathbf{v}_{ki} \cdot \sqrt{\lambda_k + k \cdot \delta} \end{bmatrix},$$

where the order of the components is according to the sorted $\lambda_j + \epsilon_j$ in ascending order. This order is reflected in some permutation matrix $\mathbf{M}$, which we will use next. Now, since we can choose δ arbitrarily low, we can choose it such that

$$\left|\left| f(\{(\mathbf{v}_{ji}, \lambda_j + j \cdot \delta)\}_{j=1}^{k}) - \mathbf{U}\mathbf{\Sigma}_i^{\frac{1}{2}}\mathbf{M} \right|\right|_{\mathrm{F}} < \epsilon,$$

for any $\epsilon > 0$. Further, f is permutation invariant and can be approximated by a DeepSet arbitrarily close. As a result, we have

$$\left|\left| \mathrm{adj}(v_i) - \mathbf{U}\mathbf{\Sigma}_i^{\frac{1}{2}}\mathbf{M} \right|\right|_{\mathrm{F}} < \epsilon,$$

for all i and an arbitrarily small $\epsilon > 0$. In matrix form, we have that

$$\left|\left| \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M} - \mathbf{Q}^A \right|\right|_{\mathrm{F}} < \epsilon.$$

Now, we need to distinguish between the non-normalized and normalized Laplacian. In the first case, we consider the graph Laplacian underlying the eigendecomposition to be non-normalized, i.e., $\mathbf{L} = \mathbf{D} - \mathbf{A}(G)$. First, we know from Lemma 17 that $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ is adjacency-identifying. Further, we know from Lemma 9 that then $\mathbf{Q}^A$ is sufficiently adjacency-identifying, since $\mathbf{Q}^A$ can approximate $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ arbitrarily close. As a result, $\mathbf{P}$ has a sufficiently adjacency-identifying sub-space and is, thus, also sufficiently adjacency-identifying.

In the second case, we consider the graph Laplacian underlying the eigendecomposition to be normalized, i.e., $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}(G)\mathbf{D}^{-\frac{1}{2}}$. Here, recall our construction for $\mathbf{P}$, namely

$$\mathbf{P}(v) = \mathsf{FFN}\Big(\begin{bmatrix} \deg(v) & \mathrm{node}(v) & \mathrm{adj}(v) \end{bmatrix}\Big)$$

or in matrix form

$$\mathbf{P} = \mathsf{FFN}\Big(\begin{bmatrix} \mathbf{Q}^D & \mathbf{Q}^N & \mathbf{Q}^A \end{bmatrix}\Big).$$

We will now use FFN to approximate the following function f, defined as

$$f\Big(\begin{bmatrix} \mathbf{Q}^D & \mathbf{Q}^N & \mathbf{Q}^A \end{bmatrix}\Big) = \begin{bmatrix} \mathbf{Q}^D & \mathbf{Q}^N & \mathbf{D}^{\frac{1}{2}}\mathbf{Q}^A \end{bmatrix}.$$

Note that a FFN can approximate such a function arbitrarily close since our domain is compact and left-multiplication by $\mathbf{D}^{\frac{1}{2}}$ is equivalent to multiplying the i-th row of $\mathbf{Q}^A$ with $\sqrt{d_i}$, where d_i is the degree of node i. Further, the i-th row of $\mathbf{Q}^D$ is an embedding $\deg(v_i)$ of d_i. We can choose this embedding to be

$$\deg(v_i) = \begin{bmatrix} \sqrt{d_i} & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^p,$$

where we write $\sqrt{d_i}$ into the first component and pad the remaining vector with zeros to fit the target dimension p of deg. Hence, with the FFN we can approximate a sub-space containing $\mathbf{D}^{\frac{1}{2}}\mathbf{Q}^A$ arbitrarily close. Further, since we already showed that $\mathbf{Q}^A$ can approximate $\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ arbitrarily close, we can thus approximate $\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ arbitrarily close. First, we know from Lemma 18 that $\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{M}$ is adjacency-identifying. Further, we know from Lemma 9 that then $\mathbf{D}^{\frac{1}{2}}\mathbf{Q}^A$ is sufficiently adjacency-identifying, since $\mathbf{D}^{\frac{1}{2}}\mathbf{Q}^A$ can approximate $\mathbf{D}^{\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{Q}^A$ arbitrarily close. As a result, $\mathbf{P}$ has a sufficiently adjacency-identifying sub-space and is, thus, also sufficiently adjacency-identifying according to Lemma 19.

Finally, for $\mathbf{Q}^D$ we need $p \geq 1$, for $\mathbf{Q}^N$ and $\mathbf{Q}^A$ we need $\frac{s}{2} \geq n$. As a result, the above statements hold for $d \geq (2n + 1)$. This concludes the proof. □

F.4. SPE

Here, we show that SPE are sufficiently node- and adjacency-identifying. We begin with useful lemma.

Lemma 21. *Let G be a graph with graph Laplacian $\mathbf{L}$ and normalized graph Laplacian $\tilde{\mathbf{L}}$. Then, if for a node-level PE $\mathbf{Q}$ with compact domain there exists matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ such that*

$$\frac{1}{\sqrt{d_k}}(\mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T = \tilde{\mathbf{L}},$$

there exists a parameterization of the structural embedding $\mathbf{P}$ with $\mathbf{Q}$ as node-level PE such that $\mathbf{P}$ is sufficiently node- and adjacency-identifying.

Proof. Recall that structural embeddings with $\mathbf{P}$ as node-level PEs are defined as

$$\mathbf{P}(v) = \mathsf{FFN}(\deg(v) + \mathbf{Q}(v)),$$

for all $v \in V(G)$. Since the domains of both deg and $\mathbf{Q}$ are compact, there exists parameters for both of these embeddings such that without loss of generality we may assume that for $v \in V(G)$,

$$\deg(v) + \mathbf{Q}(v) = \begin{bmatrix}\deg(v) & \mathbf{Q}(v)\end{bmatrix} \in \mathbb{R}^d$$

and that deg and $\mathbf{Q}$ are instead embedded into some smaller p-dimensional and s-dimensional sub-spaces, respectively, where it holds that $d = p + s$.

Next, we choose the embedding $\deg(v)$ to be

$$\deg(v) = \begin{bmatrix}\sqrt{d_v} & 0 & \dots & 0\end{bmatrix} \in \mathbb{R}^p,$$

where d_v is the degree of node v and we write $\sqrt{d_v}$ into the first component and pad the remaining vector with zeros to fit the target dimension p of deg. We will now use FFN to approximate the following function f, defined as

$$f\Big(\begin{bmatrix}\deg(v) & \mathbf{Q}(v)\end{bmatrix}\Big) = \begin{bmatrix}\deg(v) & \sqrt{d_v}\mathbf{Q}(v)\end{bmatrix}.$$

Note that a FFN can approximate such a function arbitrarily close since our domain is compact. Hence, we have that

$$\big|\big|\mathbf{P}(v) - \begin{bmatrix}\deg(v) & \sqrt{d_v}\mathbf{Q}(v)\end{bmatrix}\big|\big|_{\mathrm{F}} < \epsilon_1,$$

for all $\epsilon_1 > 0$. Further, in matrix form this becomes

$$\left|\left|\mathbf{P} - \begin{bmatrix}\begin{bmatrix}\deg(v_1)\\ \vdots \\ \deg(v_n)\end{bmatrix} & \mathbf{D}^{\frac{1}{2}}\mathbf{Q}\end{bmatrix}\right|\right|_{\mathrm{F}} < \epsilon_2,$$

for all $\epsilon_2 > 0$, since left multiplication of a matrix by $\mathbf{D}^{\frac{1}{2}}$ corresponds to an element-wise multiplication of the i-th row of $\mathbf{Q}$ with $\sqrt{d_{v_i}}$, the square-root of the degree of the i-th node v_i in an arbitrary but fixed node ordering. Now, since we have that there exists matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ such that

$$\frac{1}{\sqrt{d_k}}(\mathbf{Q}\mathbf{W}^Q)(\mathbf{Q}\mathbf{W}^K)^T = \tilde{\mathbf{L}},$$

then, for the same matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ we know that

$$\frac{1}{\sqrt{d_k}}(\mathbf{D}^{\frac{1}{2}}\mathbf{Q}\mathbf{W}^Q)(\mathbf{D}^{\frac{1}{2}}\mathbf{Q}\mathbf{W}^K)^T = \mathbf{D}^{\frac{1}{2}}\tilde{\mathbf{L}}\mathbf{D}^{\frac{1}{2}} = \mathbf{L}$$

Finally, since $\mathbf{P}$ has a sub-matrix that can approximate $\mathbf{D}^{\frac{1}{2}}\mathbf{Q}$ arbitrarily close and since, by Lemma 15, $\mathbf{D}^{\frac{1}{2}}\mathbf{Q}$ is node- and adjacency-identifying, $\mathbf{P}$ is sufficiently node- and adjacency-identifying. This shows the statement. □

We now show Theorem 8.

Theorem 22 (Theorem 8 in the main paper)**.** *Structural embeddings with SPE as node-level PE are sufficiently node- and adjacency-identifying.*

Proof. We begin by stating that the domain of SPE is compact, since eigenvectors are unit-norm and eigenvalues are bounded by twice the maximum node degree for graphs without self-loops (Anderson & Morley, 1985). Further, the domain of the structural embeddings is compact, since SPE is compact and the node degrees are finite and hence the domain of deg is compact.

Let G be a graph with (normalized or non-normalized) graph Laplacian $\mathbf{L}$. Let $\mathbf{L} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$ be the eigendecomposition of $\mathbf{L}$, where the i-th column of $\mathbf{V}$ contains the i-th eigenvector, denoted $\mathbf{v}_i$, and the i-th diagonal entry of $\mathbf{\Sigma}$ contains the i-th eigenvalue, denoted λ_i.

Recall that structural embeddings with SPE as node-level PEs are defined as

$$\mathbf{P}(v) = \mathsf{FFN}(\deg(v) + \mathrm{SPE}(v)),$$

for all $v \in V(G)$, where

$$\mathrm{SPE}(v) = \rho\Big(\begin{bmatrix}\mathbf{V}\mathrm{diag}(\phi_1(\lambda))\mathbf{V}^T & \dots & \mathbf{V}\mathrm{diag}(\phi_n(\lambda))\mathbf{V}^T\end{bmatrix}\Big)(v).$$

To show node **and** adjacency identifiability, we first replace the neural networks $\rho, \phi_1, \dots, \phi_n$ in SPE with functions permutation equivariant functions $g, h_1, \dots, h_n$ over the same domain and co-domain, respectively, and choose g and $h_1, \dots, h_n$ such that the resulting encoding, which we call f_{SPE}, is sufficiently node- and adjacency-identifying. Then, we show that ρ and $\phi_1, \dots, \phi_n$ can approximate g and $h_1, \dots, h_n$ arbitrarily close, which gives the proof statement.

We first define, for all $\ell \in [n]$,

$$h_\ell(\lambda) = \begin{bmatrix} 0 \\ \vdots \\ \lambda_\ell^{\frac{1}{2}} \\ \vdots \\ 0 \end{bmatrix},$$

such that λ_ℓ is the ℓ-th sorted eigenvalue where ties are broken arbitrarily[3] and λ_ℓ is the ℓ-th entry of $h_\ell(\lambda)$. Note that due to the sorting, h_ℓ is equivariant to permutations of the nodes.

We then define the matrix $\mathbf{M}_i$ as i-th input to g in f_{SPE}, for all $i \in [n]$. We have

$$\begin{aligned}
\mathbf{M}_i &\coloneqq \begin{bmatrix}\mathbf{V}\mathrm{diag}(h_1(\lambda))\mathbf{V}_i^T & \dots & \mathbf{V}\mathrm{diag}(h_n(\lambda))\mathbf{V}_i^T\end{bmatrix} \\
&= \begin{bmatrix} \sum_j v_{1j} \cdot h_1(\lambda)_j \cdot v_{ij} & \dots & \sum_j v_{1j} \cdot h_n(\lambda)_j \cdot v_{ij} \\ & \ddots & \\ \sum_j v_{nj} \cdot h_1(\lambda)_j \cdot v_{ij} & \dots & \sum_j v_{nj} \cdot h_n(\lambda)_j \cdot v_{ij} \end{bmatrix} \\
&= \begin{bmatrix} v_{11} \cdot \lambda_1^{\frac{1}{2}} \cdot v_{i1} & \dots & v_{1n} \cdot \lambda_n^{\frac{1}{2}} \cdot v_{in} \\ & \ddots & \\ v_{n1} \cdot \lambda_1^{\frac{1}{2}} \cdot v_{i1} & \dots & v_{nn} \cdot \lambda_n^{\frac{1}{2}} \cdot v_{in} \end{bmatrix},
\end{aligned}$$

where the last equality follows from the fact that $h_\ell(\lambda)_j \neq 0$ if and only if $\ell = j$. Now, we choose a linear set function f, satisfying

$$f(\{a \cdot x \mid x \in X\}) = a \cdot f(X),$$

for all $a \in \mathbb{R}$ and all $X \subset \mathbb{R}$, e.g., sum or mean. Since f is a function over sets, f is equivariant to the ordering of X. We now define the function g step-by-step. In g, we first apply f to the columns of $\mathbf{M}_i$ for all $i \in [n]$. To this end, let $\mathbf{M}_{ij}$ denote the *set* of entries in the j-column of matrix $\mathbf{M}_i$. Then, we define

$$\begin{aligned}
\mathbf{f}_i &= \begin{bmatrix} f(\mathbf{M}_{i1}) & \dots & f(\mathbf{M}_{in}) \end{bmatrix} \\
&= \Big[\lambda_1^{\frac{1}{2}} \cdot v_{i1} \cdot f(\{v_{o1} \mid o \in [n]\}) \quad \dots \quad \lambda_n^{\frac{1}{2}} \cdot v_{in} \cdot f(\{v_{on} \mid o \in [n]\})\Big],
\end{aligned}$$

[3] Note that despite eigenvalues with higher multiplicity, the output of h_ℓ is the same, no matter how ties are broken.

We then multiply $\mathbf{f}_i$ element-wise to each row of $\mathbf{M}_i$ and obtain a matrix

$$\begin{aligned}\mathbf{P}_i &= \mathbf{M}_i \cdot \mathbf{f}_i \\ &= \begin{bmatrix} v_{11} \cdot (\lambda_1^{\frac{1}{2}})^2 \cdot (v_{i1})^2 & \dots & v_{1n} \cdot (\lambda_1^{\frac{1}{2}})^2 \cdot (v_{in})^2 \\ & \ddots & \\ v_{n1} \cdot (\lambda_1^{\frac{1}{2}})^2 \cdot (v_{i1})^2 & \dots & v_{nn} \cdot (\lambda_n^{\frac{1}{2}})^2 \cdot (v_{in})^2 \end{bmatrix} \\ &= \begin{bmatrix} v_{11} \cdot \lambda_1 \cdot (v_{i1})^2 & \dots & v_{1n} \cdot \lambda_n \cdot (v_{in})^2 \\ & \ddots & \\ v_{n1} \cdot \lambda_1 \cdot (v_{i1})^2 & \dots & v_{nn} \cdot \lambda_n \cdot (v_{in})^2 \end{bmatrix}.\end{aligned}$$

Further, we divide each row of $\mathbf{M}_i$ element-wise by $\mathbf{f}_i$ and obtain a matrix

$$\begin{aligned}\mathbf{Q}_i &= \mathbf{M}_i / \mathbf{f}_i \\ &= \begin{bmatrix} \frac{v_{11}}{f(\{v_{o1} \mid o \in [n]\})} & \dots & \frac{v_{1n}}{f(\{v_{on} \mid o \in [n]\})} \\ & \ddots & \\ \frac{v_{n1}}{f(\{v_{o1} \mid o \in [n]\})} & \dots & \frac{v_{nn}}{f(\{v_{on} \mid o \in [n]\})} \end{bmatrix}.\end{aligned}$$

Hence, we also need to ensure that

$$f(\{v_{oj} \mid o \in [n]\}) \neq 0,$$

for all $j \in [n]$. Now, we define

$$\mathbf{P} = \sum_i \mathbf{P}_i$$

and

$$\mathbf{Q} = \sum_i \mathbf{Q}_i.$$

and have that

$$\begin{aligned}(\mathbf{P}\mathbf{Q}^T)_{k,l} &= \sum_\ell \Big(v_{k\ell} \cdot \lambda_\ell \cdot f(\{v_{o\ell} \mid o \in [n]\}) \cdot \sum_i v_{i\ell}^2 \Big) \cdot \Big(\frac{v_{l\ell}}{f(\{v_{o\ell} \mid o \in [n]\})} \Big) \\ &= \sum_\ell v_{k\ell} \cdot v_{l\ell} \cdot \lambda_\ell,\end{aligned}$$

where we recall that the eigenvectors are normalized and hence, $\sum_i v_{i\ell}^2 = 1$.

In the last step of g, we return the matrix

$$\begin{bmatrix}\mathbf{P} & \mathbf{Q}\end{bmatrix} \in \mathbb{R}^{n \times 2n}.$$

We will now show that this matrix is node- and adjacency-identifying. To this end, note that

$$(\mathbf{P}\mathbf{Q}^T)_{k,l} = (\mathbf{V}\Sigma\mathbf{V}^T)_{k,l}$$

and hence,

$$\mathbf{P}\mathbf{Q}^T = \mathbf{V}\Sigma\mathbf{V}^T = \mathbf{L}.$$

We distinguish between two cases. If $\mathbf{L}$ is the non-normalized graph Laplacian, then according to Lemma 16, the matrix

$$\begin{bmatrix}\mathbf{P} & \mathbf{Q}\end{bmatrix}$$

is node- and adjacency-identifying and hence, according to Lemma 19, f_{SPE} is node- and adjacency-identifying. Further, if $\mathbf{L}$ is the normalized graph Laplacian, according to Lemma 21, f_{SPE} is sufficiently node- and adjacency-identifying.

To summarize, we have shown that there exist functions $g, h_1, \dots, h_n$ such that if, in SPE, we replace ρ with g and ϕ_ℓ with h_ℓ, then the resulting encoding f_{SPE} is sufficiently node- and adjacency-identifying. To complete the proof, we will now show

that we can approximate $g, h_1, \ldots, h_n$ with $\rho, \phi_1, \ldots, \phi_n$ arbitrarily close. Since our domain is compact, we can use ϕ_ℓ to approximate h_ℓ arbitrarily close, i.e., we have that

$$\left|\left|\phi_\ell - h_\ell\right|\right|_{\mathrm{F}} < \epsilon,$$

for all $\epsilon > 0$. Further, g consists of a sequence of permutation equivariant steps and is thus, permutation equivariant. Since the domain of g is compact and, by construction, g and ρ have the same domain and co-domain, and since ρ can approximate any permutation equivariant function on its domain and co-domain arbitrarily close, we have that ρ can also approximate g arbitrarily close. As a result, SPE can approximate a sufficiently node- and adjacency-identifying encoding arbitrarily close, and hence, by definition, we have that SPE is also sufficiently node- and adjacency-identifying. This completes the proof. □

G. Learnable embeddings

Here, we pay more attention to learnable embeddings, which play an important role throughout our work. Although our definition of learnable embeddings is fairly abstract, learnable embeddings are very commonly used in practice, e.g., in tokenizers of language models (Vaswani et al., 2017). Since many components of graphs, such as the node labels, edge types, or node degrees are discrete, learnable embeddings are useful to embed these discrete features into an embedding space. In our work, different embeddings are typically joined via sum, e.g., in Equation (1) or Equation (4). Summing embeddings is much more convenient in practice since every embedding has the same dimension d. In contrast, joining embeddings via concatenation can lead to very large d, unless the underlying embeddings are very low-dimensional. However, in our theorems joining embeddings via concatenation is much more convenient. To bridge theory and practice in this regard, in what follows we establish under what conditions summed embeddings can express concatenated embeddings and vice versa. Specifically, we show that summed embeddings can still act as if concatenating lower-dimensional embeddings but with more flexibility in terms of joining different embeddings.

The core idea is that we show under which operations one may replace a learnable embedding e to $\mathbb{R}^d$ with a lower-dimensional learnable embedding e' to $\mathbb{R}^{d'}$ while preserving the injectivity of e. This property is useful since it allows us to, for example, add two learnable embeddings without losing expressivity. As a result, we can design a tokenizer that is practically useful without sacrificing theoretical guarantees.

We begin with a projection of a concatenation of learnable embeddings.

Lemma 23 (Projection of learnable embeddings)**.** *Let $e'_1, \ldots, e'_k$ be learnable embeddings from $\mathbb{N}$ to $\mathbb{R}^p$ for some $p \geq 1$. If $d \geq k$ then, there exists learnable embeddings $e_1, \ldots, e_k$ as well as a projection matrix $\mathbf{W} \in \mathbb{R}^{d \cdot k \times k \cdot p}$ such that for $v_1, \ldots, v_k \in \mathbb{N}$*

$$\left[e_i(v_i)\right]_{i=1}^{k} \mathbf{W} = \left[e'_i(v_i)\right]_{i=1}^{k} \in \mathbb{R}^{k \cdot p},$$

where it holds for all $v, w \in \mathbb{N}$ and all $i \in \{1, \ldots, k\}$ that

$$e_i(v) = e_i(w) \iff e'_i(v) = e'_i(w).$$

Proof. For a learnable embedding e and some $v \in \mathbb{N}$, we denote with $\mathrm{e}(v)_j$ the j-th element of the vector $\mathrm{e}(v)$. Since learnable embeddings are from $\mathbb{N}$ to $\mathbb{R}^d$, for every i we define e_i such that for every $v \in \mathbb{N}$, $\mathrm{e}_i(v)_1 = \mathrm{e}'_i(v)$ and $\mathrm{e}_i(v)_j = 0$ for $j > 0$.

Now, we define $\mathbf{W}$ as follows. For row l and column j, we set $\mathbf{W}_{lj} = 1$ if $(l-1)$ is a multiple of d, i.e., $l = 1, (d+1), 2(d+1), \ldots$. Otherwise we set $\mathbf{W}_{lj} = 0$. Intuitively, applying $\mathbf{W}$ to $\left[\mathrm{e}_i(v_i)\right]_{i=1}^{k}$ will select the first, d-th, $2d$-th, $\ldots$, kd-th, element of $\left[\mathrm{e}_i(v_i)\right]_{i=1}^{k}$, corresponding to $v_1, \ldots, v_k$, respectively, according to our construction of e_i. Now, we have that

$$\left[\mathrm{e}_i(v_i)\right]_{i=1}^{k} \mathbf{W} = \left[\mathrm{e}'_1(v_1) \ \ldots \ \mathrm{e}'_k(v_k)\right] = \left[\mathrm{e}'_i(v_i)\right]_{i=1}^{k}.$$

Further we have by construction that for all $v, w \in \mathbb{N}$ and all $i \in \{1, \ldots, k\}$

$$\mathrm{e}_i(v) = \mathrm{e}_i(w) \iff \mathrm{e}'_i(v) = \mathrm{e}'_i(w).$$

This shows the statement. □

Next, we turn to a composition of learnable embeddings, namely the structural embeddings $\mathbf{P}$, as defined in Section 3.1. In particular, we show next that a projection of multiple structural embeddings preserves node- and adjacency identifiability of the individual structural embeddings in the new embedding space.

Lemma 24 (Projection of structural embeddings)**.** *Let G be a graph with n nodes and let $\mathbf{P} \in \mathbb{R}^{n\times d}$ be structural embeddings for G. Let $k > 1$. If $d \geq k \cdot (2n+1)$ then, there exists a parameterization of $\mathbf{P}$ as well as a projection matrix $\mathbf{W} \in \mathbb{R}^{k\cdot d\cdot k\times(2n+1)}$ such that for any $v_1, \ldots, v_k \in V(G)$,*

$$\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W} = \big[\mathbf{P}'(v_i)\big]_{i=1}^{k} \in \mathbb{R}^{n\times d},$$

where $\mathbf{P}' \in \mathbb{R}^{n\times d'}$ is also a structural embedding for G and it holds that if $\mathbf{P}$ is sufficiently node- and adjacency-identifying, then so is $\mathbf{P}'$.

Proof. We know from Theorem 20 that there exists sufficiently node- and adjacency-identifying structural embeddings $\mathbf{P}'$ in $\mathbb{R}^{n\times(2d+1)}$. Since $d \geq k\cdot(2n+1)$, we define

$$\mathbf{P} = \begin{bmatrix}\mathbf{P}' & \mathbf{0}\end{bmatrix},$$

where $\mathbf{0} \in \mathbb{R}^{n\times(k-1)\cdot(2n+1)}$ is an all-zero matrix. Since $\mathbf{P}$ has a sufficiently node- and adjacency-identifying subspace $\mathbf{P}'$, by Lemma 19, $\mathbf{P}$ is sufficiently node- and adjacency-identifying. Further, by setting

$$\mathbf{W} = \mathbf{I}_{n\times k\cdot(2n+1)},$$

where $\mathbf{I}_{n\times k\cdot(2n+1)}$ are the first n rows of the $k\cdot(2n+1)$-dimensional identity matrix and $\mathbf{0} \in \mathbb{R}^{n\times(k-1)\cdot(2n+1)}$ is again an all-zero matrix.

Then, we have that for nodes $v_1, \ldots, v_k \in V(G)$,

$$\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W} = \big[\mathbf{P}'(v_i)\big]_{i=1}^{k} \in \mathbb{R}^{n\times d},$$

and $\mathbf{P}'$ is by definition sufficiently node- and adjacency-identifying, from which the statement follows. □

We continue by showing that our construction of token embeddings in Equation (1) can be equally well represented by another form that will be easier to use in proving Theorem 2.

Lemma 25. *Let G be a graph with node features $\mathbf{F} \in \mathbb{R}^{n\times d}$. For every tokenization $\mathbf{X}^{(0,1)} \in \mathbb{R}^{d}$ according to Equation (1) with (sufficiently) adjacency reconstructing structural embeddings $\mathbf{P}(v)$, there exists a parameterization of $\mathbf{X}^{(0,1)}$ such that for node $v \in V(G)$ as*

$$\mathbf{X}^{(0,1)}(v) = \begin{bmatrix}\mathbf{F}'(v) & \mathbf{0} & \mathit{deg}'(v) & \mathbf{P}'(v)\end{bmatrix},$$

with

$$\mathbf{P}'(v) = \mathit{FFN}'(\mathit{deg}'(v) + \mathit{PE}(v)),$$

such that $\mathbf{F}'(v) \in \mathbb{R}^p$, $\mathbf{0} \in \mathbb{R}^p$, $\mathit{deg}' : \mathbb{N} \to \mathbb{R}^r$ and $\mathit{FFN}' : \mathbb{R}^d \to \mathbb{R}^s$ for $d = 2p + r + s$ and where it holds for every $v, w \in V(G)$ that

$$\mathbf{F}(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w)$$

and

$$\mathit{deg}(v) = \mathit{deg}(w) \iff \mathit{deg}'(v) = \mathit{deg}'(w)$$

and

$$\mathbf{P}(v) = \mathbf{P}(w) \iff \mathbf{P}'(v) = \mathbf{P}'(w)$$

and $\mathbf{P}'(v)$ is (sufficiently) adjacency-identifying.

Proof. We set $p = {}^{(d\,-\,(2n\,+\,1))}\!/_2 - 1$, $r = 2$ and $s = (2n+1)$. Indeed, we have that

$$2p + r + s = 2({}^{(d\,-\,(2n\,+\,1))}\!/_2 - 1) + 2 + (2n+1) = d - (2n+1) + (2n+1) = d.$$

We continue by noting that the node features $\mathbf{F}$ are mapped from $\mathbb{N}$ to $\mathbb{R}^d$ and can be obtained from a learnable embedding. More importantly, we can define

$$\mathbf{F}'(v) = \begin{bmatrix}\ell(v) & 0 & \ldots & 0\end{bmatrix},$$

where $\mathbf{F}'(v) \in \mathbb{R}^d$ is another learnable embedding for which it holds that

$$\mathbf{F}(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w),$$

for all $v, w \in V(G)$.

Further, we can write

$$\deg(v) + \mathrm{PE}(v) = \begin{bmatrix}\deg'(v) & \mathrm{PE}(v)\end{bmatrix},$$

where $\deg'$ is another learnable embedding for the node degrees mapping to $\mathbb{R}^r$. Moreover, we know from Theorem 20 that there exist (sufficiently) adjacency-identifying structural embeddings of dimension $(2n+1)$. We choose the FFN in Equation (2) such that

$$\mathbf{P}(v) = \begin{bmatrix}\mathbf{0} & \deg'(v) & \mathbf{P}'\end{bmatrix},$$

where $\mathbf{0} \in \mathbb{R}^{2p}$ is an all-zero vector and

$$\mathbf{P}' = \mathsf{FFN}'(\deg'(v) + \mathrm{PE}(v)),$$

with $\mathsf{FFN}' : \mathbb{R}^d \to \mathbb{R}^s$, another FFN inside of FFN. Note that such a FFN exists without approximation. Further according to Lemma 24, $\mathbf{P}'$ is still (sufficiently) adjacency-identifying. Hence, we can write

$$\mathbf{X}^{(0,1)}(v) = \mathbf{F}'(v) + \mathbf{P}'(v) = \begin{bmatrix}\mathbf{F}'(v) & \mathbf{0} & \deg'(v) & \mathsf{FFN}'(\deg'(v) + \mathrm{PE}(v))\end{bmatrix},$$

where through the concatenation it is easy to verify that indeed $\mathbf{X}^{(0,1)}(v)$ has the desired properties, namely that

$$\mathbf{F}(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w)$$

and

$$\deg(v) = \deg(w) \iff \deg'(v) = \deg'(w)$$

and

$$\mathbf{P}(v) = \mathbf{P}(w) \iff \mathbf{P}'(v) = \mathbf{P}'(w)$$

and $\mathbf{P}'(v)$ is (sufficiently) adjacency-identifying. This shows the statement. □

Lastly, we show a similar result for Equation (4).

Lemma 26. *Let G be a graph with node features $\mathbf{F} \in \mathbb{R}^{n \times d}$. For every tokenization $\mathbf{X}^{(0,k)} \in \mathbb{R}^d$ according to Equation (4) with node- and adjacency reconstructing structural embeddings $\mathbf{P}(v)$, there exists a parameterization of $\mathbf{X}^{(0,k)}$ such that for k-tuple $\mathbf{v} = (v_1, \ldots, v_k) \in V(G)^k$,*

$$\mathbf{X}^{(0,k)}(v) = \begin{bmatrix}\mathbf{F}'(v_1) & \ldots & \mathbf{F}'(v_k) & \mathbf{0} & \mathit{deg}'(v_1) & \ldots & \mathit{deg}'(v_k) & \mathbf{P}'(v_1) & \ldots & \mathbf{P}'(v_k) & \mathit{atp}'(\mathbf{v})\end{bmatrix},$$

with

$$\mathbf{P}'(v) = \mathit{FFN}'(\mathit{deg}'(v) + \mathit{PE}(v)),$$

such that $\mathbf{F}'(v) \in \mathbb{R}^p$, $\mathbf{0} \in \mathbb{R}^{k^2 \cdot p}$, $\mathit{deg}' : \mathbb{N} \to \mathbb{R}^r$, $\mathit{atp}' : \mathbb{N} \to \mathbb{R}^o$ and $\mathbf{P}' \in \mathbb{R}^s$ for $d = k^2 \cdot p + k \cdot p + k \cdot r + k \cdot s + o$ and where it holds for every $v, w \in V(G)$,

$$\mathbf{F}'(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w)$$

and

$$\mathit{deg}(v) = \mathit{deg}(w) \iff \mathit{deg}'(v) = \mathit{deg}'(w)$$

and

$$\mathbf{P}(v) = \mathbf{P}(w) \iff \mathbf{P}'(v) = \mathbf{P}'(w)$$

and $\mathbf{P}'(v)$ is node and adjacency-identifying and for every $\mathbf{v}, \mathbf{w} \in V(G)^k$

$$\mathit{atp}(\mathbf{v}) = \mathit{atp}(\mathbf{w}) \iff \mathit{atp}'(\mathbf{v}) = \mathit{atp}'(\mathbf{w}).$$

Proof. We set $p = 1$, $r = 2$, $o = 1$ and $s = \frac{d-k^2-3k-1}{k}$. Indeed, we have that

$$k^2 \cdot p + k \cdot p + k \cdot r + k \cdot s + o = k^2 + 3k + d - k^2 - 3k - 1 + 1 = d.$$

Recall Equation (4) as

$$\mathbf{X}^{(0,k)}(\mathbf{v}) = \big[\mathbf{F}(v_i)\big]_{i=1}^{k}\mathbf{W}^F + \big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W}^P + \mathrm{atp}(\mathbf{v}),$$

where $\mathbf{W}^F \in \mathbb{R}^{d\cdot k\times d}$ and $\mathbf{W}^P \in \mathbb{R}^{d\cdot k\times d}$ are projection matrices, $\mathbf{P}$ are structural embeddings and $\mathbf{v} = (v_1, \ldots, v_k)$ is a k-tuple.

We continue by noting that the node features $\mathbf{F}$ are mapped from $\mathbb{N}$ to $\mathbb{R}^d$ and can be obtained from a learnable embedding. More importantly, we can define $\mathbf{F}'(v) = \ell(v)$ for which it clearly holds that

$$\mathbf{F}(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w),$$

for all $v, w \in V(G)$. Invoking Lemma 23, there exists learnable embeddings $\mathbf{F}$ and a projection matrix $\mathbf{W}^{F,*} \in \mathbb{R}^{k\cdot d\times k\cdot p}$ such that

$$\big[\mathbf{F}(v_i)\big]_{i=1}^{k}\mathbf{W}^F = \begin{bmatrix}\mathbf{F}'(v_1) & \ldots & \mathbf{F}'(v_k)\end{bmatrix} \in \mathbb{R}^{k\cdot p}.$$

Note that since we do not have an assumption on d, we can make d arbitrarily large to ensure that $d \geq k$. Hence, by defining

$$\mathbf{W}^F = \begin{bmatrix}\mathbf{W}^{F,*} & \mathbf{0}\end{bmatrix},$$

where $\mathbf{0} \in \mathbb{R}^{k\cdot d\times(d-k\cdot p)}$ is an all-zero matrix, we can write

$$\big[\mathbf{F}(v_i)\big]_{i=1}^{k}\mathbf{W}^F = \begin{bmatrix}\mathbf{F}'(v_1) & \ldots & \mathbf{F}'(v_k) & \mathbf{0}\end{bmatrix} \in \mathbb{R}^{d},$$

where $\mathbf{0} \in \mathbb{R}^{d-k\cdot p}$ is an all-zero vector.

Moreover, we know from Theorem 7 that there exists adjacency-identifying structural embeddings of dimension $(2n+1)$. We choose the FFN in Equation (2) such that

$$\mathbf{P}(v) = \begin{bmatrix}\mathbf{0} & \deg'(v) & \mathbf{P}'(v)\end{bmatrix},$$

where $\mathbf{0} \in \mathbb{R}^{k^2\cdot p+k\cdot p}$ is an all-zero vector and

$$\mathbf{P}'(v) = \mathsf{FFN}'(\deg'(v) + \mathrm{PE}(v)),$$

with $\mathsf{FFN}' : \mathbb{R}^d \to \mathbb{R}^s$, another FFN inside of FFN. Note that such a FFN exists without approximation. Further, according to Lemma 24, $\mathbf{P}'$ is still sufficiently adjacency-identifying, and there exists a projection matrix $\mathbf{W}^{P,*} \in \mathbb{R}^{k\cdot d\times d-o}$ such that

$$\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W}^{P,*} = \begin{bmatrix}\mathbf{0} & \deg'(v_1) & \mathbf{P}'(v_1) & \ldots & \mathbf{0} & \deg'(v_k) & \mathbf{P}'(v_k)\end{bmatrix} \in \mathbb{R}^{d-o}.$$

But then, there also exists a permutation matrix $\mathbf{Q} \in \{0,1\}^{d-o\times d-o}$ such that

$$\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W}^{P,*}\mathbf{Q} = \begin{bmatrix}\mathbf{0} & \deg'(v_1) & \ldots & \deg'(v_k) & \mathbf{P}'(v_1) & \ldots & \mathbf{P}'(v_k)\end{bmatrix} \in \mathbb{R}^{d-o},$$

where we simply re-order the entries of $\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W}^{P,*}$. Note that now, $\mathbf{0} \in \mathbb{R}^{k^2\cdot p+k\cdot p}$, as we grouped the zero-vectors of $\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W}^{P,*}$ into a single zero-vector of size $k^2 \cdot p + k \cdot p$.

Note that since we do not have an assumption on d, we can make d arbitrarily large to ensure that $d \geq k \cdot (2n+1)$. Hence, by defining

$$\mathbf{W}^P = \begin{bmatrix}\mathbf{W}^{P,*}\mathbf{Q} & \mathbf{0}\end{bmatrix},$$

where $\mathbf{0} \in \mathbb{R}^o$ is an all-zero vector, we can write

$$\big[\mathbf{P}(v_i)\big]_{i=1}^{k}\mathbf{W}^P = \begin{bmatrix}\mathbf{0} & \deg'(v_1) & \ldots & \deg'(v_k) & \mathbf{P}'(v_1) & \ldots & \mathbf{P}'(v_k) & \mathbf{0}\end{bmatrix} \in \mathbb{R}^{d},$$

where $\mathbf{P}'(v_i) \in \mathbb{R}^s$. Further, since $\mathbf{P}$ is by assumption sufficiently node- and adjacency-identifying, Lemma 24 guarantees that then so is $\mathbf{P}'$.

Finally, for tuple $\mathbf{v} \in V(G)^k$, we set

$$\text{atp}(\mathbf{v}) = \begin{bmatrix}\mathbf{0} & \text{atp}'(\mathbf{v})\end{bmatrix},$$

where $\text{atp}'(\mathbf{v}) = \text{atp}(\mathbf{v})$ and we recall that atp maps to $\mathbb{N}$ and $\mathbf{0} \in \mathbb{R}^{d-1}$ is an all-zero vector.

Then, we can write

$$\begin{aligned}\mathbf{X}^{(0,k)}(\mathbf{v}) &= \begin{bmatrix}\mathbf{F}(v_i)\end{bmatrix}_{i=1}^{k}\mathbf{W}^F + \begin{bmatrix}\mathbf{P}(v_i)\end{bmatrix}_{i=1}^{k}\mathbf{W}^P + \text{atp}(\mathbf{v}) \\ &= \begin{bmatrix}\mathbf{F}'(v_1) & \dots & \mathbf{F}'(v_k) & \mathbf{0} & \deg'(v_1) & \dots & \deg'(v_k) & \mathbf{P}'(v_1) & \dots & \mathbf{P}'(v_k) & \text{atp}'(\mathbf{v})\end{bmatrix},\end{aligned}$$

where it holds for every $v, w \in V(G)$,

$$\mathbf{F}'(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w)$$

and

$$\deg(v) = \deg(w) \iff \deg'(v) = \deg'(w)$$

and

$$\mathbf{P}(v) = \mathbf{P}(w) \iff \mathbf{P}'(v) = \mathbf{P}'(w)$$

where $\mathbf{P}'(v)$ is node and adjacency-identifying and for every $\mathbf{v}, \mathbf{w} \in V(G)^k$

$$\text{atp}(\mathbf{v}) = \text{atp}(\mathbf{w}) \iff \text{atp}'(\mathbf{v}) = \text{atp}'(\mathbf{w}).$$

This shows the statement. □

H. Expressivity of 1-GT

Here, we prove Theorem 2 in the main paper. We first restate Theorem VIII.4 of (Grohe, 2021), showing that a simple GNN can simulate the 1-WL.

Lemma 27 (Grohe (2021), Theorem VIII.4)**.** *Let $G = (V(G), E(G), \ell)$ be a graph with n nodes with adjacency matrix $\mathbf{A}(G)$ and node feature matrix $\mathbf{X}^{(0)} := \mathbf{F} \in \mathbb{R}^{n \times d}$ consistent with ℓ. Further, assume a GNN that for each layer, $t > 0$, updates the vertex feature matrix*

$$\mathbf{X}^{(t)} := \mathsf{FFN}\Big(\mathbf{X}^{(t-1)} + 2\mathbf{A}(G)\mathbf{X}^{(t-1)}\Big).$$

Then, for all $t \geq 0$, there exists a parameterization of FFN *such that*

$$C^1_t(v) = C^1_t(w) \iff \mathbf{X}^{(t)}(v) = \mathbf{X}^{(t)}(w),$$

for all vertices $v, w \in V(G)$.

We now give the proof for a slight generalization of Theorem 2. Specifically, we relax the adjacency-identifying condition to *sufficiently* adjacency-identifying.

Theorem 28 (Generalization of Theorem 2 in main paper)**.** *Let $G = (V(G), E(G), \ell)$ be a labeled graph with n nodes and $\mathbf{F} \in \mathbb{R}^{n \times d}$ be a node feature matrix consistent with ℓ. Let C^1_t denote the coloring function of the 1-WL at iteration t. Then, for all iterations $t \geq 0$, there exists a parametrization of the 1-GT with sufficiently adjacency-identifying structural embeddings, such that*

$$C^1_t(v) = C^1_t(w) \iff \mathbf{X}^{(t,1)}(v) = \mathbf{X}^{(t,1)}(w),$$

for all nodes $v, w \in V(G)$.

Proof. According to Lemma 25, there exists a parameterization of $\mathbf{X}^{(0,1)}$ such that for each $v \in V(G)$,

$$\mathbf{X}^{(0,1)}(v) = \mathbf{F}'(v) + \mathbf{P}'(v) = \begin{bmatrix}\mathbf{F}'(v) & \mathbf{0} & \deg'(v) & \mathbf{P}'(v)\end{bmatrix},$$

where it holds that

$$\mathbf{F}(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w) \tag{17}$$

and

$$\deg(v) = \deg(w) \iff \deg'(v) = \deg'(w)$$

and

$$\mathbf{P}(v) = \mathbf{P}(w) \iff \mathbf{P}'(v) = \mathbf{P}'(w)$$

and $\mathbf{P}'(v)$ is adjacency-identifying. Further, $\mathbf{F}'(v), \mathbf{0} \in \mathbb{R}^p$, $\deg'(v) \in \mathbb{R}^r$ and $\mathbf{P}'(v) \in \mathbb{R}^s$, for some choice of p, r, s where $d = 2p + r + s$, specified in Lemma 25. We will use this parameterization for $\mathbf{X}^{(0,1)}$ throughout the rest of the proof.

We will now prove the statement by induction over t. Since by definition

$$C^1_0(v) = C^1_0(w) \iff \mathbf{F}(v) = \mathbf{F}(w),$$

the base case follows from Equation (17). We let $\mathbf{F}^{(t)}(v)$ denote the representation of the color of node v at iteration t. Initially, we set $\mathbf{F}^{(t)}(v) = \mathbf{F}'(v)$. Further, we define the matrix $\mathbf{D}_{\text{emb}} \in \mathbb{R}^{n \times r}$ such that for the i-th row of $\mathbf{D}_{\text{emb},i} = \deg'(v_i)$, where v_i is the i-th node in a fixed but arbitrary node ordering. Hence, we can write

$$\mathbf{X}^{(0,1)}(v) = \begin{bmatrix}\mathbf{F}^{(0)}(v) & \mathbf{0} & \deg'(v) & \mathbf{P}'(v)\end{bmatrix},$$

for all $v \in V(G)$ and in matrix form

$$\mathbf{X}^{(0,1)} = \begin{bmatrix}\mathbf{F}^{(0)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{P}'\end{bmatrix}.$$

Now, assume that the statement holds up to iteration $t-1$. For the induction, we show

$$C^1_t(v) = C^1_t(w) \iff \mathbf{F}^{(t)}(v) = \mathbf{F}^{(t)}(w). \tag{18}$$

That is, we compute the 1-WL-equivalent aggregation using the node color representations $\mathbf{F}^{(t)}$ and use the remaining columns in $\mathbf{X}^{(0,1)}$ to keep track of degree and structural embeddings. Clearly, if Equation (18) holds for all t, then the statement follows. Thereto, we aim to update the vertex representations such that

$$\mathbf{X}^{(t,1)} = \begin{bmatrix}\mathbf{F}^{(t)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{P}'\end{bmatrix}, \tag{19}$$

where, following Lemma 27,

$$\mathbf{F}^{(t)} \coloneqq \mathsf{FFN}\Big(\mathbf{F}^{(t-1)} + 2\mathbf{A}(G)\mathbf{F}^{(t-1)}\Big). \tag{20}$$

Hence, it remains to show that our graph transformer layer can update vertex representations according to Equation (19). To this end, we will require only a single transformer head in each layer. Specifically, we want to compute

$$h_1(\mathbf{X}^{(t-1,1)}) = \begin{bmatrix}\mathbf{0} & \mathbf{D}^{-1}\mathbf{A}(G))\mathbf{F}^{(t-1)} & \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{21}$$

where $\mathbf{D}^{-1}$ denotes the inverse of the degree matrix and $\mathbf{I}$ denotes the identity matrix with n rows. Note that since we only have one head, the head dimension $d_v = d$. We begin by re-stating Equation (8) with expanded sub-matrices and then derive the instances necessary to obtain the head Equation (21). We re-state projection weights $\mathbf{W}^Q$ and $\mathbf{W}^K$ with expanded sub-matrices as

$$\mathbf{W}^Q = \begin{bmatrix}\mathbf{W}^Q_1 \\ \mathbf{W}^Q_2 \\ \mathbf{W}^Q_3 \\ \mathbf{W}^Q_4\end{bmatrix}$$

and

$$\mathbf{W}^K = \begin{bmatrix}\mathbf{W}^K_1 \\ \mathbf{W}^K_2 \\ \mathbf{W}^K_3 \\ \mathbf{W}^K_4\end{bmatrix},$$

where $\mathbf{W}_1^Q, \mathbf{W}_1^K \in \mathbb{R}^{p\times d}, \mathbf{W}_2^Q, \mathbf{W}_2^K \in \mathbb{R}^{p\times d}, \mathbf{W}_3^Q, \mathbf{W}_3^K \in \mathbb{R}^{r\times d}, \mathbf{W}_4^Q, \mathbf{W}_4^K \in \mathbb{R}^{s\times d}$. We then define

$$\mathbf{Z}^{(t-1)} \coloneqq \frac{1}{\sqrt{d_k}}(\mathbf{X}^{(t-1,1)}\mathbf{W}^Q)(\mathbf{X}^{(t-1,1)}\mathbf{W}^K)^T = \frac{1}{\sqrt{d_k}}(\mathbf{X}^{(t-1,1)}\begin{bmatrix}\mathbf{W}_1^Q\\ \mathbf{W}_2^Q\\ \mathbf{W}_3^Q\\ \mathbf{W}_4^Q\end{bmatrix})(\mathbf{X}^{(t-1,1)}\begin{bmatrix}\mathbf{W}_1^K\\ \mathbf{W}_2^K\\ \mathbf{W}_3^K\\ \mathbf{W}_4^K\end{bmatrix})^T,$$

where

$$\mathbf{X}^{(t-1,1)} = \begin{bmatrix}\mathbf{F}^{(t-1)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{P}'\end{bmatrix},$$

by the induction hypothesis. By setting $\mathbf{W}_1^Q, \mathbf{W}_2^Q, \mathbf{W}_3^Q, \mathbf{W}_1^K, \mathbf{W}_2^K, \mathbf{W}_3^K$ to zero, we have

$$\mathbf{Z}^{(t-1)} = \frac{1}{\sqrt{d_k}}(\mathbf{P}'\mathbf{W}_4^Q)(\mathbf{P}'\mathbf{W}_4^K)^T.$$

Now, let

$$\tilde{\mathbf{P}} \coloneqq \frac{1}{\sqrt{d_k}}(\mathbf{P}'\mathbf{W}_P^Q)(\mathbf{P}'\mathbf{W}_P^K)^T,$$

for some weight matrices $\mathbf{W}_P^Q$ and $\mathbf{W}_P^K$. We know from Lemma 12 that, since $\mathbf{P}'$ is sufficiently adjacency reconstructing, there exists $\mathbf{W}_P^Q$ and $\mathbf{W}_P^K$ such that by setting $\mathbf{W}_4^Q = b \cdot \mathbf{W}_P^Q$ and $\mathbf{W}_4^K = \mathbf{W}_P^K$, we have

$$\mathbf{Z}^{(t-1)} = b \cdot \tilde{\mathbf{P}},$$

where for all $\varepsilon > 0$, there exists a $b > 0$, such that

$$\left\|\mathsf{softmax}\Big(\mathbf{Z}^{(t-1)}\Big) - \mathbf{D}^{-1}\mathbf{A}(G)\right\|_F < \varepsilon.$$

Hence, by choosing a large enough b, we can approximate the matrix $\mathbf{D}^{-1}\mathbf{A}(G)$ arbitrarily close. In the following, for clarity of presentation, we assume $\mathsf{softmax}\Big(\mathbf{Z}^{(t)}\Big) = \mathbf{D}^{-1}\mathbf{A}(G)$ although we only approximate it arbitrarily close. However, by choosing ε small enough, we can still approximate the matrix $\mathbf{X}^{(t,1)}$, see below, arbitrarily close.

We now again expand sub-matrices of $\mathbf{W}^V$ and express Equation (8) as

$$h_i(\mathbf{X}^{(t-1,1)}) = \mathsf{softmax}\Big(\mathbf{Z}^{(t-1)}\Big)\mathbf{X}^{(t-1,1)}\begin{bmatrix}\mathbf{W}_1^V\\ \mathbf{W}_2^V\\ \mathbf{W}_3^V\\ \mathbf{W}_4^V\end{bmatrix},$$

where $\mathbf{W}_1^V \in \mathbb{R}^{p\times d}, \mathbf{W}_2^V \in \mathbb{R}^{p\times d}, \mathbf{W}_3^V \in \mathbb{R}^{r\times d}, \mathbf{W}_4^V \in \mathbb{R}^{s\times d}$. By setting

$$\mathbf{W}_1^V = \begin{bmatrix}\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\end{bmatrix}$$
$$\mathbf{W}_2^V = \mathbf{W}_3^V = \mathbf{W}_4^V = \mathbf{0},$$

where $\mathbf{I} \in \mathbb{R}^{p\times p}$, we can approximate

$$h_1(\mathbf{X}^{(t-1,1)}) = \begin{bmatrix}\mathbf{0} & \mathbf{D}^{-1}\mathbf{A}(G)\mathbf{F}^{(t-1)} & \mathbf{0} & \mathbf{0}\end{bmatrix}$$

arbitrarily close. We now conclude our proof as follows. Recall that the transformer computes the final representation $\mathbf{X}^{(t,1)}$ as

$$\begin{aligned}\mathbf{X}^{(t,1)} &= \mathsf{FFN}_{\text{final}}\left(\mathbf{X}^{(t-1,1)} + h_1(\mathbf{X}^{(t-1,1)})\mathbf{W}^O\right)\\ &= \mathsf{FFN}_{\text{final}}\left(\begin{bmatrix}\mathbf{F}^{(t-1)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix} + \begin{bmatrix}\mathbf{0} & \mathbf{D}^{-1}\mathbf{A}(G)\mathbf{F}^{(t-1)} & \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{W}^O\right)\\ &\underset{\mathbf{W}^O \coloneqq \mathbf{I}}{=} \mathsf{FFN}_{\text{final}}\left(\begin{bmatrix}\mathbf{F}^{(t-1)} & \mathbf{D}^{-1}\mathbf{A}(G)\mathbf{F}^{(t-1)} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix}\right),\end{aligned}$$

for some $\mathsf{FFN}_{\text{final}}$. We now show that there exists an $\mathsf{FFN}_{\text{final}}$ such that

$$\mathbf{X}^{(t,1)} = \mathsf{FFN}_{\text{final}}\left(\begin{bmatrix}\mathbf{F}^{(t-1)} & \mathbf{D}^{-1}\mathbf{A}(G)\mathbf{F}^{(t-1)} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{F}^{(t)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix},$$

which then implies the proof statement. To this end, we show that there exists a composition of functions $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} \circ f_{\times 2}$ such that

$$f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} \circ f_{\times 2}\left(\begin{bmatrix}\mathbf{F}^{(t-1)} & \mathbf{D}^{-1}\mathbf{A}(G)\mathbf{F}^{(t-1)} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{F}^{(t)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix},$$

where the functions are applied independently to each row. We denote

$$\mathbf{X}^{(t-1)}_{\text{upd}} \coloneqq \begin{bmatrix}\mathbf{F}^{(t-1)} & \mathbf{D}^{-1}\mathbf{A}(G)\mathbf{F}^{(t-1)} & \mathbf{D}_{\text{emb}} & \mathbf{U}\end{bmatrix} \in \mathbb{R}^{n \times d}$$

and obtain for node $v \in V(G)$,

$$\mathbf{X}^{(t-1)}_{\text{upd}}(v) = \begin{bmatrix}\mathbf{F}^{(t-1)}(v) & \sum_{w \in N(v)} \frac{1}{|N(v)|} \cdot \mathbf{F}^{(t-1)}(w) & \deg'(v) & \mathbf{P}'(v)\end{bmatrix} \in \mathbb{R}^d.$$

We can now define

$$\deg'(v) = \begin{bmatrix}|N(v)| & 0\end{bmatrix} \in \mathbb{R}^2.$$

Since our domain is compact, there exist choices of $f_{\text{FFN}}, f_{\text{add}}, f_{\text{deg}}, f_{\times 2}$ such that $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} \circ f_{\times 2} : \mathbb{R}^d \to \mathbb{R}^d$ is continuous. As a result, $\mathsf{FFN}_{\text{final}}$ can approximate $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} \circ f_{\times 2}$ arbitrarily close.

Concretely, we define

$$\begin{aligned} &f_{\times 2}\left(\begin{bmatrix}\mathbf{F}^{(t-1)}(v) & \sum_{w \in N(v)} \frac{1}{|N(v)|} \cdot \mathbf{F}^{(t-1)}(w) & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix}\right) \\ &\quad = \begin{bmatrix}\mathbf{F}^{(t-1)}(v) & 2\sum_{w \in N(v)} \frac{1}{|N(v)|} \cdot \mathbf{F}^{(t-1)}(w) & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix}, \end{aligned}$$

where $f_{\times 2}$ multiplies the second column with 2.

Next, we define

$$\begin{aligned} &f_{\text{deg}}\left(\begin{bmatrix}\mathbf{F}^{(t-1)}(v) & 2\sum_{w \in N(v)} \frac{1}{|N(v)|} \cdot \mathbf{F}^{(t-1)}(w) & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix}\right) \\ &\quad = \begin{bmatrix}\mathbf{F}^{(t-1)}(v) & 2\sum_{w \in N(v)} \frac{1}{|N(v)|} \cdot \mathbf{F}^{(t-1)}_j \cdot |N(v)| & |N(v)| & \mathbf{P}'(v)\end{bmatrix} \\ &\quad = \begin{bmatrix}\mathbf{F}^{(t-1)}(v) & 2\sum_{w \in N(v)} \mathbf{F}^{(t-1)}(w) & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix} \end{aligned}$$

where f_{deg} multiplies the second column with the degree of node v in the third column.

Next, we define

$$\begin{aligned} &f_{\text{add}}\left(\begin{bmatrix}\mathbf{F}^{(t-1)}(v) & 2\sum_{w \in N(v)} \mathbf{F}^{(t-1)}(w) & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix}\right) \\ &\quad = \begin{bmatrix}\mathbf{F}^{(t-1)}(v) + 2\sum_{w \in N(v)} \mathbf{F}^{(t-1)}(w) & \mathbf{0} & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix}, \end{aligned}$$

where we add the second column to the first column and set the elements of the second column to zero.

Finally, we define

$$\begin{aligned} &f_{\text{FFN}}\left(\begin{bmatrix}\mathbf{F}^{(t-1)}(v) + 2\sum_{w \in N(v)} \mathbf{F}^{(t-1)}(w) & \mathbf{0} & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix}\right) \\ &\quad = \begin{bmatrix}\mathsf{FFN}\left(\mathbf{F}^{(t-1)}(v) + 2\sum_{w \in N(v)} \mathbf{F}^{(t-1)}(w)\right) & \mathbf{0} & |N(v)| & 0 & \mathbf{P}'(v)\end{bmatrix} \\ &\quad = \begin{bmatrix}\mathbf{F}^{(t)}(v) & \mathbf{0} & \deg'(v) & \mathbf{P}'(v)\end{bmatrix}, \end{aligned}$$

where FFN denotes the FFN in Equation (20), from which the last equality follows. Clearly, applying $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} \circ f_{\times 2}$ to each row i of $\mathbf{X}_{\text{upd}}^{(t-1)}$ results in

$$f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} \circ f_{\times 2}\left(\mathbf{X}_{\text{upd}}^{(t-1)}\right) = \begin{bmatrix}\mathbf{F}^{(t)} & \mathbf{0} & \mathbf{D}_{\text{emb}} & \mathbf{P}'(v)\end{bmatrix},$$

from which our proof follows. □

I. Expressivity of higher-order transformers

Here, we prove Theorem 4, Theorem 5, and Theorem 6 in Section 3.2 in the main paper. To this end, we first construct new higher-order GNNs aligned with the k-WL, δ-k-WL, δ-k-LWL, and (k, s)-WL, respectively. We will use these higher-order GNNs as an intermediate step to show the expressivity results for the k-GT and (k, s)-GT.

I.1. Higher-order GNNs

Here, we generalize the GNN from Grohe (2021) to higher orders k. Higher-order GNNs with the same expressivity have been proposed in prior works by Azizian & Lelarge (2021) and Morris et al. (2019). However, our GNNs have a special form that can be computed by a transformer. To this end, we extend Theorem VIII.4 in Grohe (2021) to the k-WL. Specifically, for each $k > 1$ we devise neural architectures with k-WL expressivity. Afterwards, we show that transformers can simulate these architectures.

Formally, let $S \subset \mathbb{N}$ be a finite subset. First, we show that multisets over S can be injectively mapped to a value in the closed interval $(0, 1)$, which is a variant of Lemma VIII.5 in Grohe (2021). Here, we outline a streamlined version of its proof, highlighting the key intuition behind representing multisets as m-ary numbers. Let $M \subseteq S$ be a multiset with multiplicities $a_1, \ldots, a_k$ and distinct k values. We define the *order* of the multiset as $\sum_{i=1}^{k} a_i$. We can write such a multiset as a sequence $x^{(1)}, \ldots, x^{(l)}$ where l is the order of the multiset. Note that the order of the sequence is arbitrary and that for $i \neq j$ it is possible to have $x^{(i)} = x^{(j)}$. We call such a sequence an M-sequence of length l. We now prove the following, a slight variation of Grohe (2021).

Lemma 29. *For a finite $m \in \mathbb{N}$, let $M \subseteq S$ be a multiset of order $m - 1$ and let $x_i \in S$ denote the i-th number in a fixed but arbitrary ordering of S. Given a mapping $g \colon S \to (0, 1)$ where*

$$g(x_i) \coloneqq m^{-i},$$

and an M-sequence of length l given by $x^{(1)}, \ldots, x^{(l)}$ with positions $i^{(1)}, \ldots, i^{(l)}$ in S, the sum

$$\sum_{j \in [l]} g(x^{(j)}) = \sum_{j \in [l]} m^{-i^{(j)}}$$

is unique for every unique M.

Proof. By assumption, let $M \subseteq S$ denote a multiset of order $m - 1$. Further, let $x^{(1)}, \ldots, x^{(l)} \in M$ be an M-sequence with $i^{(1)}, \ldots, i^{(l)}$ in S. Given our fixed ordering of the numbers in S we can equivalently write $M = ((a_1, x_1), \ldots, (a_n, x_n))$, where a_i denotes the multiplicity of i-th number in M with position i from our ordering over S. Note that for a number m^{-i} there exists a corresponding m-ary number written as

$$0.0 \ldots \underbrace{1}_{i} \ldots$$

Then the sum,

$$\begin{aligned}\sum_{j \in [l]} g(x^{(j)}) &= \sum_{j \in [l]} m^{-i^{(j)}} \\ &= \sum_{i \in S} a_i m^{-i} \in (0, 1)\end{aligned}$$

and in m-ary representation

$$0.a_1 \dots a_n.$$

Note that $a_i = 0$ if and only if there exists no j such that $i^{(j)} = i$. Since the order of M is $m-1$, it holds that $a_i < m$. Hence, it follows that the above sum is unique for each unique multiset M, implying the result. □

Recall that $S \subseteq \mathbb{N}$ and that we fixed an arbitrary ordering over S. Intuitively, we use the finiteness of S to map each number therein to a fixed digit of the numbers in $(0, 1)$. The finite m ensures that at each digit, we have sufficient "bandwidth" to encode each a_i. Now that we have seen how to encode multisets over S as numbers in $(0, 1)$, we review some fundamental operations about the m-ary numbers defined above. We will refer to decimal numbers m^{-i} as *corresponding* to an m-ary number

$$0.0 \dots \underbrace{1}_{i} \dots,$$

where the i-th digit after the decimal point is 1 and all other digits are 0, and vice versa.

To begin with, addition between decimal numbers implements *counting* in m-ary notation, i.e.,

$$m^{-i} + m^{-j} \text{ corresponds to } 0.0 \dots \underbrace{1}_{i} \dots \underbrace{1}_{j} \dots,$$

for digit positions $i \neq j$ and

$$m^{-i} + m^{-j} \text{ corresponds to } 0.0 \dots \underbrace{2}_{i=j} \dots,$$

otherwise. We used counting in the proof of the previous result to represent the multiplicities of a multiset. Next, multiplication between decimal numbers implements *shifting* in m-ary notation, i.e.,

$$m^{-i} \cdot m^{-j} \text{ corresponds to } 0.0 \dots \underbrace{1}_{i+j} \dots.$$

Shifting further applies to general decimal numbers in $(0, 1)$. Let $x \in (0, 1)$ correspond to an m-ary number with l digits,

$$0.a_1 \dots a_l.$$

Then,

$$m^{-i} \cdot x \text{ corresponds to } 0.0 \dots 0 \underbrace{a_1 \dots a_l}_{i+1,\dots,i+l}.$$

We continue by deriving a neural architecture simulating the k-WL.

Proposition 30. *Let $G = (V(G), E(G), \ell)$ be a n-order (node-)labeled graph. Then for all $t \geq 1$, there exists a function $g^{(t)}$ and scalars $\beta_1, \dots, \beta_k$ with*

$$g^{(t)}(\mathbf{u}) \coloneqq \mathsf{FFN}\Big(g^{(t-1)}(\mathbf{u}) + \sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} g^{(t-1)}(\phi_j(\mathbf{u}, w))\Big),$$

where $g^{(0)}(\mathbf{u})$ is initialized consistent ℓ consistent with the atomic type, such that for all $\mathbf{u}, \mathbf{v} \in V(G)^k$

$$C^k_t(\mathbf{u}) = C^k_t(\mathbf{v}) \iff g^{(t)}(\mathbf{u}) = g^{(t)}(\mathbf{v}). \tag{22}$$

Proof. Before we start, let us recall the relabeling computed by the k-WL for a k-tuple $\mathbf{u}$ as

$$\mathsf{RELABEL}\Big(C^k_t(\mathbf{u}), M^{(t)}_1(\mathbf{u}), \dots, M^{(t)}_k(\mathbf{u})\Big),$$

with

$$M^{(t)}_j(\mathbf{u}) \coloneqq \{\!\{C^k_{t-1}(\phi_j(\mathbf{u}, w)) \mid w \in V(G)\}\!\}.$$

To show our result, we show that there exist scalars $\beta_1, \ldots, \beta_k$ such that the m-ary representations computed for $C^k_t(\mathbf{u})$ and $M^{(t)}_1, \ldots, M^{(t)}_k$ are pairwise unique. To this end, we show that a weighted sum can represent multiset counting in different exponent ranges of m-ary numbers in $(0, 1)$. We then simply invoke Lemma 29 to show that we can map each unique tuple $(C^k_t(\mathbf{u}), M^{(t)}_1(\mathbf{u}), \ldots, M^{(t)}_k(\mathbf{u}))$ to a unique number in $(0, 1)$. Finally, the FFN will be responsible for the relabeling.

Each possible color in C^k_t is a unique number in $[n^k]$ as the maximum possible number of unique colors in C^k_t is n^k. We then fix an arbitrary ordering over the $[n^k]$.

We show the statement via induction over t. Let $m > 0$ such that $m - 1$ is the order of the multiset $M^{(0)}_j(\mathbf{u})$. Note that this is the same m for each $\mathbf{u}$. For a k-tuple $\mathbf{u}$ with initial color $C^k_0(\mathbf{u})$ at position i, we choose $g^{(0)}$ such as to approximate m^{-i} arbitarily close, i.e.,

$$\left|\left|g^{(0)}(\mathbf{u}) - m^{-i}\right|\right|_{\mathrm{F}} < \epsilon,$$

for an arbitrarily small $\epsilon > 0$. By choosing ϵ small enough, we have that $g^{(0)}(\mathbf{u})$ is unique for every unique position i and hence Equation (22) holds for all pairs of k-tuples for $t = 0$. Note that by construction, $i \leq n^k$ and hence, for tuple $\mathbf{u}$ at position i, the m-ary number corresponding to m^{-i} is non-zero in at most the first n^k digits.

For the inductive case, we assume that

$$C^k_{t-1}(\mathbf{u}) = C^k_{t-1}(\mathbf{v}) \iff g^{(t-1)}(\mathbf{u}) = g^{(t-1)}(\mathbf{v}).$$

Further, we assume that for tuple $\mathbf{u}$ at position i,

$$\left|\left|g^{(t-1)}(\mathbf{u}) - m^{-i}\right|\right|_{\mathrm{F}} < \epsilon,$$

for an arbitrarily small $\epsilon > 0$. Let now $\mathbf{u} \in V(G)^k$. We say that the j-neighbors of $\mathbf{u}$ w.r.t. w have indices $l^{(1)}(w), \ldots, l^{(k)}(w)$ in our ordering. Then, $\sum_{w \in V(G)} m^{-l^{(j)}(w)}$ is unique for each unique

$$M^{(t)}_j(\mathbf{u}) = \{\!\{C^k_t(\phi_j(\mathbf{u}, w)) \mid w \in V(G)\}\!\}, \tag{23}$$

according to Lemma 29.

We now obtain a unique value for the tuple

$$\left(M^{(t)}_1(\mathbf{u}), \ldots, M^{(t)}_k(\mathbf{u})\right),$$

by setting

$$\beta_j \coloneqq m^{-(n^k)\cdot j},$$

for $j \in [k]$. Specifically, let $a_1, \ldots, a_{n^k}$ denote the multiplicities of multiset $M^{(t)}_j(\mathbf{u})$ and let

$$0.a_1 \ldots a_{n^k},$$

denote the m-ary number that corresponds to the sum

$$\sum_{w \in V(G)} m^{-l^{(j)}(w)}.$$

Note that since each $m^{-l^{(j)}(w)}$ corresponds to a color under k-WL at iteration $t - 1$, all digits after the n^k-th digit are zero. Then, multiplying the above sum with β_j results in a shift in m-ary notation and, hence, for the m-ary number that corresponds to the term

$$\beta_j \cdot \sum_{w \in V(G)} m^{-l^{(j)}(w)}, \tag{24}$$

we can write

$$0.0 \ldots \underbrace{0}_{n^k \cdot j} \quad \underbrace{a_1 \ldots a_{n^k}}_{n^k \cdot j+1, \ldots, n^k \cdot j + n^k} \quad \underbrace{0}_{n^k \cdot (j+1)+1} \quad \ldots 0.$$

As a result, for all $l \neq j$, the non-zero digits of the m-ary number that corresponds to

$$\beta_l \cdot \sum_{w \in V(G)} m^{-l^{(j)}(w)}$$

do not collide with the non-zero digits of the output of Equation (24) and hence, the sum

$$\sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} m^{-l^{(j)}(w)}, \tag{25}$$

is unique for each unique tuple

$$\Big(M_1^{(t)}(\mathbf{u}), \ldots, M_k^{(t)}(\mathbf{u})\Big).$$

Let $\mathbf{u}_i$ be the i-th tuple in our fixed but arbitrary ordering. Then, $g^{(t-1)}(\mathbf{u})$ approximates the number m^{-i} arbitrarily close. Note that by the induction hypothesis, the m-ary number that corresponds to m^{-i} is non-zero in at most the first n^k digits, as n^k is the maximum possible number of colors attainable under k-WL. Since the smallest shift in Equation (25) is by n^k (for $j = 1$) and since the m-ary number that corresponds to m^{-i} is non-zero in at most the first n^k digits, the sum of m^{-i} and Equation (25) have no intersecting non-zero digits. As a result,

$$m^{-i} + \sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} m^{-l^{(j)}(w)} \tag{26}$$

is unique for each tuple

$$\Big(C_t^k(\mathbf{u}), M_1^{(t)}(\mathbf{u}), \ldots, M_k^{(t)}(\mathbf{u})\Big).$$

Now, by the induction hypothesis, we can approximate each m^{-i} with $g^{(t-1)}(\mathbf{u}_i)$ arbitrarily close. Further, we can approximate each $m^{-l^{(j)}(w)}$ with $g^{(t-1)}(\phi_j(\mathbf{u}, w))$ arbitrarily close. As a result, we can approximate Equation (26) with

$$g^{(t-1)}(\mathbf{u}) + \sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} g^{(t-1)}(\phi_j(\mathbf{u}, w))$$

arbitrarily close, i.e.,

$$\Big|\Big|\Big(g^{(t-1)}(\mathbf{u}) + \sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} g^{(t-1)}(\phi_j(\mathbf{u}, w))\Big) - \Big(m^{-i} + \sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} m^{-l^{(j)}(w)}\Big)\Big|\Big|_{\mathrm{F}} < \epsilon,$$

for an arbitrarily small ϵ. Hence, we by choosing ϵ small enough,

$$g^{(t-1)}(\mathbf{u}) + \sum_{j \in [k]} \beta_j \cdot \sum_{w \in V(G)} g^{(t-1)}(\phi_j(\mathbf{u}, w))$$

is also unique for each unique tuple

$$\Big(C_t^k(\mathbf{u}), M_1^{(t)}(\mathbf{u}), \ldots, M_k^{(t)}(\mathbf{u})\Big).$$

Finally, since for finite graphs of order n there exists only a finite number of such tuples, there exists a continuous function mapping the output of Equation (26) to m^{-i} where i is the position of the color $C_t^k(\mathbf{u})$ in $[n^k]$, for all $\mathbf{u} \in V(G)^k$. We approximate this function with FFN arbitrarily close and obtain

$$||g^{(t)}(\mathbf{u}) - m^{-i}|| < \epsilon,$$

for an arbitrarily small $\epsilon > 0$. This concludes the proof. □

We finish with a few Corollaries regarding variants of the k-WL.

Corollary 31. *Let $G = (V(G), E(G), \ell)$ be a n-order (node-)labeled graph. Then for each k-tuple $\mathbf{u} \coloneqq (u_1, \dots u_k)$ and each $t > 0$, we can equivalently express the coloring of the δ-k-WL as $C^{\delta,k}_t(\mathbf{u}) \coloneqq$*

$$\mathsf{RELABEL}\Big(C^{\delta,k}_{t-1}(\mathbf{u}), \{\!\{C^{\delta,k}_{t-1}(\phi_1(\mathbf{u}, w)) \mid w \in N(u_1)\}\!\}, \{\!\{C^{\delta,k}_{t-1}(\phi_1(\mathbf{u}, w)) \mid w \in V(G) \setminus N(u_1)\}\!\}, \dots, \\ \{\!\{C^{\delta,k}_{t-1}(\phi_k(\mathbf{u}, w)) \mid w \in N(u_k)\}\!\}, \{\!\{C^{\delta,k}_{t-1}(\phi_k(\mathbf{u}, w)) \mid w \in V(G) \setminus N(u_k)\}\!\}\Big).$$

With the above statement, we can directly derive a δ-variant of Proposition 30. To this end, let $\Delta_j(\mathbf{u})$ denote the set of vertices adjacent to the j-th node in $\mathbf{u}$.

Corollary 32. *Let $G = (V(G), E(G), \ell)$ be a n-order (vertex-)labeled graph and assume a vertex feature matrix $\mathbf{F} \in \mathbb{R}^{n \times d}$ that is consistent with ℓ. Then for all $t \geq 1$, there exists a function $g^{(t)}$ and scalars $\beta_1, \dots, \beta_{2k}$ with*

$$g^{(t)}(\mathbf{u}) \coloneqq \mathsf{FFN}\Big(g^{(t-1)}(\mathbf{u}) + \sum_{j \in [k]} \Big(\alpha_j \cdot \sum_{w \in \Delta_j(\mathbf{u})} g^{(t-1)}(\phi_j(\mathbf{u}, w)) \; + \; \beta_j \cdot \sum_{w \in V(G) \setminus \Delta_j(\mathbf{u})} g^{(t-1)}(\phi_j(\mathbf{u}, w)\Big)\Big),$$

such that for all $\mathbf{u}, \mathbf{v} \in V(G)^k$

$$C^{k,M}_t(\mathbf{u}) = C^{k,M}_t(\mathbf{v}) \iff g^{(t)}(\mathbf{u}) = g^{(t)}(\mathbf{v}). \tag{27}$$

The proof is the same as for Proposition 30. Further, we can also recover the δ-k-LWL variant of Proposition 30 by setting $\beta_j = 0$. Lastly, we again recover Proposition 30 by setting $\alpha_j = \beta_j$.

I.2. Higher-order pure transformers

Here, we use the results from the previous section to show the expressivity results for Theorem 4, Theorem 5 and Theorem 6 in the main paper. In fact, we prove a slightly stronger result than Theorem 4 from which then also Theorem 5 and Theorem 6 will follow directly.

Theorem 33 (Generalization of Theorem 4 in main paper)**.** *Let $G = (V(G), E(G), \ell)$ be a labeled graph with n nodes and $k \geq 2$ and $\mathbf{F} \in \mathbb{R}^{n \times d}$ be a node feature matrix consistent with ℓ. Let C^k_t denote the coloring function of the k-WL, δ-k-WL, or δ-k-LWL at iteration t. Then for all iterations $t \geq 0$, there exists a parametrization of the k-GT such that*

$$C^k_t(\mathbf{v}) = C^k_t(\mathbf{w}) \iff \mathbf{X}^{(t,k)}(\mathbf{v}) = \mathbf{X}^{(t,k)}(\mathbf{w})$$

for all k-tuples $\mathbf{v}$ and $\mathbf{w} \in V(G)^k$. If C^k_t is the coloring function of the k-WL, it suffices that the structural embeddings of k-GT are sufficiently node-identifying. Otherwise, we require the structural embeddings of k-GT to be both sufficiently node and adjacency-identifying.

Proof. We begin by recalling Equation (4) as

$$\mathbf{X}^{(0,k)}(\mathbf{v}) \coloneqq \big[\mathbf{F}(v_i)\big]_{i=1}^k \mathbf{W}^F + \big[\mathbf{P}(v_i)\big]_{i=1}^k \mathbf{W}^P + \text{atp}(\mathbf{v}),$$

where $\mathbf{P}$ are structural embeddings and atp is a learnable embedding of the atomic type. We further know from Lemma 26 that we can write

$$\mathbf{X}^{(0,k)}(\mathbf{v}) = \big[\mathbf{F}'(v_1) \;\; \dots \;\; \mathbf{F}'(v_k) \;\; \mathbf{0} \;\; \deg'(v_1) \;\; \dots \;\; \deg'(v_k) \;\; \mathbf{P}'(v_1) \;\; \dots \;\; \mathbf{P}'(v_k) \;\; \text{atp}'(\mathbf{v})\big], \tag{28}$$

where it holds for every $v, w \in V(G)$,

$$\mathbf{F}(v) = \mathbf{F}(w) \iff \mathbf{F}'(v) = \mathbf{F}'(w)$$

and

$$\deg(v) = \deg(w) \iff \deg'(v) = \deg'(w)$$

and

$$\mathbf{P}(v) = \mathbf{P}(w) \iff \mathbf{P}'(v) = \mathbf{P}'(w)$$

where $\mathbf{P}'(v)$ is sufficiently node and adjacency-identifying and for every $\mathbf{v}, \mathbf{w} \in V(G)^k$

$$\text{atp}(\mathbf{v}) = \text{atp}(\mathbf{w}) \iff \text{atp}'(\mathbf{v}) = \text{atp}'(\mathbf{w}).$$

Further, $\mathbf{F}'(v) \in \mathbb{R}^p$, $\mathbf{0} \in \mathbb{R}^{k^2 \cdot p}$, $\deg' \colon \mathbb{N} \to \mathbb{R}^r$, $\text{atp}' \colon \mathbb{N} \to \mathbb{R}^o$ and $\mathbf{P}' \in \mathbb{R}^s$, for some choice of p, r, s, o where $d = k^2 \cdot p + k \cdot p + k \cdot r + k \cdot s + o$, as specified in Lemma 26. We will use this parameterization for $\mathbf{X}^{(0,k)}$ throughout the rest of the proof.

We prove our statement by induction over iteration t. For the base case, notice that the initial color of a tuple $\mathbf{v}$ depends on the atomic type and the node labeling. In Equation (28), we encode the atomic type with $\text{atp}'(\mathbf{v})$ and the node labels by concatenating the features $\mathbf{F}'(v_1), \dots, \mathbf{F}'(v_k)$ of the k nodes $v_1, \dots, v_k$ in $\mathbf{v}$. The concatenation of both node labels and atomic type is clearly injective, and so

$$C^k_0(\mathbf{v}) = C^k_0(\mathbf{w}) \iff \mathbf{X}^{(0,k)}(\mathbf{v}) = \mathbf{X}^{(0,k)}(\mathbf{w})$$

for any two $\mathbf{v}, \mathbf{w} \in V(G)^k$.

Before continuing with the induction, we define some additional notation. Throughout the induction, we will denote the color representation of a tuple $\mathbf{v}$ at iteration t as $\mathbf{F}^{(t)}(\mathbf{v})$. Further, we initially define $\mathbf{F}^{(0)}(\mathbf{v}) = \big[\mathbf{F}'(v_i)\big]^k_{i=1}$, $\deg'(\mathbf{v}) = \big[\deg'(v_i)\big]^k_{i=1}$ and $\mathbf{P}'(\mathbf{v}) = \big[\mathbf{P}'(v_i)\big]^k_{i=1}$. Hence, we can rewrite

$$\mathbf{X}^{(0,k)}(\mathbf{v}) = \big[\mathbf{F}^{(0)}(\mathbf{v}) \quad \mathbf{0} \quad \deg'(\mathbf{v}) \quad \mathbf{P}'(\mathbf{v}) \quad \text{atp}'(\mathbf{v})\big].$$

For the induction step, we show

$$C^k_t(\mathbf{v}) = C^k_t(\mathbf{w}) \iff \mathbf{X}^{(t,k)}(\mathbf{v}) = \mathbf{X}^{(t,k)}(\mathbf{w}), \tag{29}$$

that is we compute the k-WL-, δ-k-LWL, or δ-k-WL-equivalent aggregation. Clearly, if Equation (29) holds for all t, then the proof statement follows. Thereto, we show that the standard transformer updates the tuple representation of tuple $\mathbf{u}_i = (u_1, \dots, u_k)$, following Corollary 32, as

$$\mathbf{X}^{(t,k)}(\mathbf{u}_i) = \Big[\mathsf{FFN}\Big(\mathbf{F}^{(t-1)}(\mathbf{u}_i) + H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i))\Big) \quad \mathbf{0} \quad \deg'(\mathbf{u}_i) \quad \mathbf{P}'(\mathbf{u}_i) \quad \text{atp}'(\mathbf{u}_i)\Big], \tag{30}$$

where

$$H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) \coloneqq \sum_{j \in [k]} \Big(\alpha_j \cdot \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)) + \beta_j \cdot \sum_{w \in V(G) \setminus \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w))\Big),$$

for $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \mathbb{R}$, for $j \in [k]$ that we pick such that $H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i))$ is equivalent to the neural architecture in Corollary 32. Note that we obtain the k-WL for $\alpha_j = \beta_j$ for all $j \in [k]$. Then,

$$H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \sum_{j \in [k]} \Big(\beta_j \cdot \sum_{w \in V(G)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w))\Big).$$

Further, we obtain δ-k-LWL for $\beta_j = 0$ for all $j \in [k]$. Then,

$$H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \sum_{j \in [k]} \Big(\alpha_j \cdot \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w))\Big).$$

Hence, it remains to show that the standard transformer layer can update tuple representations according to Equation (30). To this end, we will require $2k$ transformer heads $h^1_1, \dots h^1_k, h^2_1, \dots h^2_k$ in each layer. Specifically, in the first k heads, we want to compute

$$h^1_j(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \alpha_j \cdot \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)). \tag{31}$$

In the remaining k heads, we want to compute

$$h^2_j(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \beta_j \cdot \sum_{w \in V(G) \setminus \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)). \tag{32}$$

In both of the above cases, for a head h^γ_j, $\gamma \in \{-1, 1\}$ denotes the type of head, and $j \in [k]$ denotes the j-neighbors the head aggregates over. For the head dimension, we set $d_v = d$.

For each j, recall the definition of the standard transformer head at tuple $\mathbf{u}_i$ at position i as

$$h_j^\gamma(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \mathsf{softmax}\Big(\begin{bmatrix}\mathbf{Z}_{i1}^{(t-1,\gamma)} & \dots & \mathbf{Z}_{in^k}^{(t-1,\gamma)}\end{bmatrix}\Big)\begin{bmatrix}\mathbf{X}^{t-1}(\mathbf{u}_1) & \dots & \mathbf{X}^{t-1}(\mathbf{u}_{n^k})\end{bmatrix}^T\mathbf{W}^V,$$

where

$$\mathbf{Z}_{il}^{(t-1,\gamma)} \coloneqq \frac{1}{\sqrt{d_k}}(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)\mathbf{W}^{Q,\gamma})(\mathbf{X}^{(t-1,k)}(\mathbf{u}_l)\mathbf{W}^{K,\gamma})^T \in \mathbb{R} \tag{33}$$

is the unnormalized attention score between tuples $\mathbf{u}_i$ and $\mathbf{u}_l$, with

$$\mathbf{X}^{(t-1,k)}(\mathbf{u}_i) = \begin{bmatrix}\mathbf{F}^{(t-1)}(\mathbf{u}_i) & \mathbf{0} & \deg'(\mathbf{u}_i) & \mathbf{P}'(\mathbf{u}_i) & \mathrm{atp}'(\mathbf{u}_i)\end{bmatrix}$$

for all $i \in [n^k]$ and

$$\mathbf{W}^{Q,\gamma} = \begin{bmatrix}\mathbf{W}_F^Q\\ \mathbf{W}_Z^Q\\ \mathbf{W}_D^Q\\ \mathbf{W}_P^{Q,\gamma}\\ \mathbf{W}_A^Q\end{bmatrix}$$

$$\mathbf{W}^{K,\gamma} = \begin{bmatrix}\mathbf{W}_F^K\\ \mathbf{W}_Z^K\\ \mathbf{W}_D^K\\ \mathbf{W}_P^{K,\gamma}\\ \mathbf{W}_A^K\end{bmatrix}$$

$$\mathbf{W}^{V} = \begin{bmatrix}\mathbf{W}_F^V\\ \mathbf{W}_Z^V\\ \mathbf{W}_D^V\\ \mathbf{W}_P^V\\ \mathbf{W}_A^V\end{bmatrix},$$

where

$$\begin{aligned}\mathbf{F}^{(t-1)}(\mathbf{u}_i) &\text{ is projected by } \mathbf{W}_F^Q, \mathbf{W}_F^K, \mathbf{W}_F^V,\\ \mathbf{0} &\text{ is projected by } \mathbf{W}_Z^Q, \mathbf{W}_Z^K, \mathbf{W}_Z^V,\\ \deg'(\mathbf{u}_i) &\text{ is projected by } \mathbf{W}_D^Q, \mathbf{W}_D^K, \mathbf{W}_D^V,\\ \mathbf{P}'(\mathbf{u}_i) &\text{ is projected by } \mathbf{W}_P^{Q,\gamma}, \mathbf{W}_P^{K,\gamma}, \mathbf{W}_P^V,\\ \mathrm{atp}'(\mathbf{u}_i) &\text{ is projected by } \mathbf{W}_A^Q, \mathbf{W}_A^K, \mathbf{W}_A^V.\end{aligned}$$

Note that only sub-matrices $\mathbf{W}_P^{Q,\gamma}$ and $\mathbf{W}_P^{K,\gamma}$ are different for different γ. We now specify projection matrices $\mathbf{W}^{Q,\gamma}$, $\mathbf{W}^{K,\gamma}$ and $\mathbf{W}^V$ in a way that allows the attention head h_j^γ to dynamically recover the j-neighborhood adjacency as well as the adjacency between j-neighbors in the attention matrix. To this end, in heads h_j^1 and h_j^2 we set

$$\begin{aligned}\mathbf{W}_F^Q = \mathbf{W}_F^K &= \mathbf{0}\\ \mathbf{W}_Z^Q = \mathbf{W}_Z^K = \mathbf{W}_Z^V &= \mathbf{0}\\ \mathbf{W}_P^V &= \mathbf{0}\\ \mathbf{W}_D^Q = \mathbf{W}_D^K = \mathbf{W}_D^V &= \mathbf{0}\\ \mathbf{W}_A^Q = \mathbf{W}_A^K = \mathbf{W}_A^V &= \mathbf{0}.\end{aligned}$$

The remaining non-zero sub-matrices are $\mathbf{W}_F^V, \mathbf{W}_P^{Q,\gamma}, \mathbf{W}_P^{K,\gamma}$, which we will define next.

We begin by defining $\mathbf{W}_P^{Q,\gamma}$ and $\mathbf{W}_P^{K,\gamma}$. Specifically, we want to choose $\mathbf{W}_P^{Q,\gamma}$ and $\mathbf{W}_P^{K,\gamma}$ such that the attention matrix in head h_j^γ can approximate the generalized adjacency matrix $\mathbf{A}^{(k,j,\gamma)}$ in Equation (11). To this end, we simply invoke Lemma 13, guaranteeing that there exists $\mathbf{W}_P^{Q,\gamma}$ and $\mathbf{W}_P^{K,\gamma}$ such that that for each $\epsilon > 0$ and each $\gamma \in \{-1, 1\}$,

$$\left\| \mathsf{softmax}\Big(\begin{bmatrix} \mathbf{Z}_{i1}^{(t-1,\gamma)} & \dots & \mathbf{Z}_{in^k}^{(t-1,\gamma)} \end{bmatrix} \Big) - \tilde{\mathbf{A}}_i^{(k,j,\gamma)} \right\|_F < \varepsilon,$$

i.e., we can approximate the generalized adjacency matrix arbitrarily close for each $k > 1, j \in [k]$ and $\gamma \in \{-1, 1\}$. In the following, for clarity of presentation, we assume

$$\mathsf{softmax}\Big(\begin{bmatrix} \mathbf{Z}_{i1}^{(t-1,\gamma)} & \dots & \mathbf{Z}_{in^k}^{(t-1,\gamma)} \end{bmatrix} \Big) = \tilde{\mathbf{A}}_i^{(k,j,\gamma)}$$

although we only approximate it arbitrarily close. However, by choosing ε small enough, we can still approximate the matrix $\mathbf{X}^{(t,k)}$, see below, arbitrarily close. We now set

$$\mathbf{W}^V = \begin{bmatrix} \mathbf{I}_{n \times kp} & \mathbf{0} \end{bmatrix},$$

where $\mathbf{I}_{n \times kp}$ denotes the first n rows of the kp-by-kp identity matrix if $kp > n$ or else the first kp columns of the n-by-n identity matrix and $\mathbf{0} \in \mathbb{R}^{n \times d - kp}$ is an all-zero matrix. The above yields

$$h_j^\gamma(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \frac{1}{d_{ij}^\gamma} \begin{bmatrix} \tilde{\mathbf{A}}_{i1}^{(k,j,\gamma)} & \dots & \tilde{\mathbf{A}}_{in^k}^{(k,j,\gamma)} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_1) \\ \vdots \\ \mathbf{F}^{(t-1)}(\mathbf{u}_{n^k}) \end{bmatrix}$$

$$= \frac{1}{d_{ij}^\gamma} \cdot \begin{cases} \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)) & \gamma = 1 \\ \sum_{w \in V(G) \setminus \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)) & \gamma = -1 \end{cases},$$

where

$$d_{ij}^\gamma = \begin{cases} d_{ij} & \gamma = 1 \\ n - d_{ij} & \gamma = -1 \end{cases}$$

with d_{ij} the degree of node u_j in k-tuple $\mathbf{u}_i = (u_1, \dots, u_k)$.

We now conclude our proof as follows. Recall that the standard transformer layer computes the final representation $\mathbf{X}^{(t,k)}(\mathbf{u}_i)$ as

$$\mathbf{X}^{(t,k)}(\mathbf{u}_i) = \mathsf{FFN}\left(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i) + \begin{bmatrix} h_1^1(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \\ \vdots \\ h_k^1(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \\ h_1^2(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \\ \vdots \\ h_k^2(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \end{bmatrix}^T \mathbf{W}^O \right).$$

We can now satisfy the requirements in Equation (31) and Equation (32) as follows.

Recall that for the first k heads we set $\gamma = 1$, in the remaining k heads we set $\gamma = -1$. Further, since we have $2k$ heads, $\mathbf{W}^O \in \mathbb{R}^{2k \cdot d_v \times d}$. We set

$$\mathbf{W}_{ij}^O = \begin{cases} \alpha_j & kp < i = j \leq k + kp \\ \beta_j & k + kp < i = j \leq 2k + kp \\ 0 & \text{else,} \end{cases}$$

i.e., we leave the first kp diagonal elements zero and then fill the next $2k$ diagonal elements of $\mathbf{W}^O$ with the α_j and β_j,

respectively. All other elements are 0. We now obtain

$$\begin{bmatrix} h_1^1(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \\ \vdots \\ h_k^1(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \\ h_1^2(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \\ \vdots \\ h_k^2(\mathbf{X}^{(t-1,k)})(\mathbf{u}_i) \end{bmatrix}^T \mathbf{W}^O = \begin{bmatrix} \frac{\alpha_1}{d_{i1}} \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\alpha_k}{d_{ik}} \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \frac{\beta_1}{n-d_{i1}} \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\beta_k}{n-d_{ik}} \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \end{bmatrix}^T \mathbf{W}^O$$

$$= \begin{bmatrix} \mathbf{0} \\ \frac{\alpha_1}{d_{i1}} \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\alpha_k}{d_{ik}} \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \frac{\beta_1}{n-d_{i1}} \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\beta_k}{n-d_{ik}} \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \mathbf{0} \end{bmatrix}^T,$$

where the first zero vector is in $\mathbb{R}^{kp}$ and the second zero vector is in $\mathbb{R}^{d-k(r+s)+o}$. We now define the vector

$$\tilde{\mathbf{X}}(\mathbf{u}_i) = \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) \\ \mathbf{0} \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \mathrm{atp}'(\mathbf{u}_i) \end{bmatrix}^T + \begin{bmatrix} \mathbf{0} \\ \frac{\alpha_1}{d_{i1}} \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\alpha_k}{d_{ik}} \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \frac{\beta_1}{n-d_{i1}} \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\beta_k}{n-d_{ik}} \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \mathbf{0} \end{bmatrix}^T$$

$$= \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) \\ \frac{\alpha_1}{d_{i1}} \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\alpha_k}{d_{ik}} \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \frac{\beta_1}{n-d_{i1}} \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\beta_k}{n-d_{ik}} \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \mathrm{atp}'(\mathbf{u}_i) \end{bmatrix}^T \in \mathbb{R}^d.$$

To simplify terms, we additionally define

$$\tilde{\mathbf{F}}^{\alpha}(\mathbf{u}_i) \coloneqq \begin{bmatrix} \frac{\alpha_1}{d_{i1}} \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\alpha_k}{d_{ik}} \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \end{bmatrix}^T$$

and

$$\tilde{\mathbf{F}}^{\beta}(\mathbf{u}_i) \coloneqq \begin{bmatrix} \frac{\beta_1}{n-d_{i1}} \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \frac{\beta_k}{n-d_{ik}} \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \end{bmatrix}^T$$

and consequently, we can write

$$\tilde{\mathbf{X}}(\mathbf{u}_i) = \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) \\ \tilde{\mathbf{F}}^{\alpha}(\mathbf{u}_i) \\ \tilde{\mathbf{F}}^{\beta}(\mathbf{u}_i) \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \text{atp}'(\mathbf{u}_i) \end{bmatrix}^T$$

We additionally define

$$\deg'(u_j) = \begin{bmatrix} d_{ij} & n - d_{ij} \end{bmatrix} \in \mathbb{R}^2,$$

where d_{ij} is again the degree of node u_j in k-tuple $\mathbf{u}_i = (u_1, \ldots, u_k)$. Note that $\tilde{\mathbf{X}}(\mathbf{u}_i)$ represents all the information we feed into the final FFN. Specifically, we obtain the updated representation of tuple $\mathbf{u}_i$ as

$$\mathbf{X}^{(t,k)}(\mathbf{u}_i) = \mathsf{FFN}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big).$$

We now show that there exists an FFN such that

$$\begin{aligned} \mathbf{X}^{(t,k)} &= \mathsf{FFN}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big) \\ &= \begin{bmatrix} \mathbf{F}^{(t)}(\mathbf{u}_i) & \mathbf{0} & \deg'(\mathbf{u}_i) & \mathbf{P}'(\mathbf{u}_i) & \text{atp}'(\mathbf{u}_i) \end{bmatrix}, \end{aligned}$$

which then implies the proof statement. It is worth to pause at this point and remind ourselves what each element in $\tilde{\mathbf{X}}(\mathbf{u}_i)$ represents. To this end, we use `PyTorch`-like array slicing, i.e., for a vector $\mathbf{x}$, $\mathbf{x}[a : b]$ corresponds to the sub-vector of length $b - a$ for $b > a$ containing the $(a + 1)$-st to b-th element of $\mathbf{x}$. E.g., for a vector $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)^T$, $\mathbf{x}[1 : 4] = (x_2, x_3, x_4)^T$. Now, we interpret $\tilde{\mathbf{X}}(\mathbf{u}_i)$ by its sub-vectors. Concretely,

1. $\tilde{\mathbf{X}}(\mathbf{u}_i)[0 : kp] \in \mathbb{R}^{kp}$ corresponds to the previous color representation $\mathbf{F}^{(t-1)}(\mathbf{u}_i)$ of tuple $\mathbf{u}_i$.
2. $\tilde{\mathbf{F}}^{\alpha}(\mathbf{u}_i)[(j-1)kp : jkp] \in \mathbb{R}^{kp}$ corresponds to the degree-normalized sum over adjacent j-neighbors $\frac{\alpha_j}{d_j} \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w))$.
3. $\tilde{\mathbf{F}}^{\beta}(\mathbf{u}_i)[(j-1)kp : jkp] \in \mathbb{R}^{kp}$ corresponds to the degree-normalized sum over non-adjacent j-neighbors $\frac{\beta_j}{n-d_j} \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w))$.
4. $\deg'(\mathbf{u}_i)[2(j-1)] = d_{ij}$, where d_{ij} is the degree of node u_j in the k-tuple $\mathbf{u}_i = (u_1, \ldots, u_k)$.
5. $\deg'(\mathbf{u}_i)[2(j-1)+1] = n - d_{ij}$, where d_{ij} is the degree of node u_j in the k-tuple $\mathbf{u}_i = (u_1, \ldots, u_k)$.

To this end, we show that there exists a sequence of functions $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}}$ such that

$$\begin{aligned} \mathbf{X}^{(t,k)} &= f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big) \\ &= \begin{bmatrix} \mathbf{F}^{(t)}(\mathbf{u}_i) & \mathbf{0} & \deg'(\mathbf{u}_i) & \mathbf{P}'(\mathbf{u}_i) & \text{atp}'(\mathbf{u}_i) \end{bmatrix}, \end{aligned}$$

where the functions are applied independently to each row.

Since our domain is compact, there exist choices of $f_{\text{FFN}}, f_{\text{add}}, f_{\text{deg}}$ such that $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}} : \mathbb{R}^d \to \mathbb{R}^d$, in which case $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}}$ is continuous. As a result, FFN can approximate $f_{\text{FFN}} \circ f_{\text{add}} \circ f_{\text{deg}}$ arbitrarily close.

Concretely, we define

$$f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big) = \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) \\ d_{i1} \cdot \frac{\alpha_1}{d_{i1}} \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ d_{ik} \cdot \frac{\alpha_k}{d_{ik}} \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ (n - d_{i1}) \cdot \frac{\beta_1}{n - d_{i1}} \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ (n - d_{ik}) \cdot \frac{\beta_k}{n - d_{ik}} \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \text{atp}'(\mathbf{u}_i) \end{bmatrix}^T,$$

where for each $j \in [k]$, f_{deg} multiplies

1. $\tilde{\mathbf{F}}^{\alpha}(\mathbf{u}_i)[(j-1)kp : jkp]$ with $\deg'(\mathbf{u}_i)[2(j-1)]$.
2. $\tilde{\mathbf{F}}^{\beta}(\mathbf{u}_i)[(j-1)kp : jkp]$ with $\deg'(\mathbf{u}_i)[2(j-1)+1]$.

We then define

$$\mathbf{F}^{\alpha}(\mathbf{u}_i) \coloneqq \begin{bmatrix} \alpha_1 \cdot \sum_{w \in \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \alpha_k \cdot \sum_{w \in \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \end{bmatrix}^T$$

and

$$\mathbf{F}^{\beta}(\mathbf{u}_i) \coloneqq \begin{bmatrix} \beta_1 \cdot \sum_{w \in V(G) \setminus \Delta_1(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_1(\mathbf{u}_i, w)) \\ \vdots \\ \beta_k \cdot \sum_{w \in V(G) \setminus \Delta_k(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_k(\mathbf{u}_i, w)) \end{bmatrix}^T$$

and consequently, we can write

$$f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big) = \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) \\ \mathbf{F}^{\alpha}(\mathbf{u}_i) \\ \mathbf{F}^{\beta}(\mathbf{u}_i) \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \text{atp}'(\mathbf{u}_i) \end{bmatrix}^T.$$

Next, we define

$$f_{\text{add}}\Big(f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big)\Big) = \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) + \sum_{j \in [k]} (\mathbf{F}^{\alpha}(\mathbf{u}_i)[(j-1)kp : jkp] + \mathbf{F}^{\beta}(\mathbf{u}_i)[(j-1)kp : jkp]) \\ \mathbf{0} \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \text{atp}'(\mathbf{u}_i) \end{bmatrix}^T,$$

where f_{add} sums up $\mathbf{F}^{(t-1)}(\mathbf{u}_i)$ with $\mathbf{F}^{\alpha}(\mathbf{u}_i)[(j-1)kp : jkp]$ and $\mathbf{F}^{\beta}(\mathbf{u}_i)[(j-1)kp : jkp]$ for each j. Afterwards, f_{add} sets

$$f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big)[kp : 2k^2p + kp] = \mathbf{0} \in \mathbb{R}^{2k^2p}.$$

Now, finally, note from the definition of $\mathbf{F}^{\alpha}(\mathbf{u}_i)$ and $\mathbf{F}^{\beta}(\mathbf{u}_i)$ that

$$\sum_{j \in [k]} (\mathbf{F}^{\alpha}(\mathbf{u}_i)[(j-1)kp : jkp] + \mathbf{F}^{\beta}(\mathbf{u}_i)[(j-1)kp : jkp]) = H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)),$$

where we recall that we defined

$$H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) = \frac{1}{n} \cdot \sum_{j \in [k]} \Big(\alpha_j \cdot \sum_{w \in \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)) + \beta_j \cdot \sum_{w \in V(G) \setminus \Delta_j(\mathbf{u}_i)} \mathbf{F}^{(t-1)}(\phi_j(\mathbf{u}_i, w)) \Big).$$

Hence, we have that

$$f_{\text{add}}\Big(f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big) = \begin{bmatrix} \mathbf{F}^{(t-1)}(\mathbf{u}_i) + H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i)) \\ \mathbf{0} \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \text{atp}'(\mathbf{u}_i) \end{bmatrix}^T.$$

Finally, we define

$$\begin{aligned} f_{\text{FFN}}\Big(f_{\text{add}}\Big(f_{\text{deg}}\Big(\tilde{\mathbf{X}}(\mathbf{u}_i)\Big)\Big) &= \begin{bmatrix} \mathsf{FFN}\Big(\mathbf{F}^{(t-1)}(\mathbf{u}_i) + H(\mathbf{X}^{(t-1,k)}(\mathbf{u}_i))\Big) \\ \mathbf{0} \\ \deg'(\mathbf{u}_i) \\ \mathbf{P}'(\mathbf{u}_i) \\ \text{atp}'(\mathbf{u}_i) \end{bmatrix}^T \\ &= \begin{bmatrix} \mathbf{F}^{(t)}(v) & \mathbf{0} & \deg'(v) & \mathbf{P}'(v) \end{bmatrix}, \end{aligned}$$

where FFN denotes the FFN in Equation (30), from which the last equality follows. Thank you for sticking around until the end. This completes the proof. □

The above proof shows that the k-$\texttt{GT}$ can simulate the δ-k-WL. Since the δ-k-WL is strictly more expressive than the k-WL (Morris et al., 2020), Theorem 5 follows directly from Theorem 4.

Further, the above proof shows that the k-$\texttt{GT}$ can simulate the δ-k-LWL. If we simulate the δ-k-LWL with the k-$\texttt{GT}$ while only using token embeddings for tuples in $V(G)^k_s$, Theorem 6 directly follows.